Semiannual Status Report

NAG 5-1612

Submitted to
Instrument Division
Engineering Directorate
Goddard Space Flight Center
Greenbelt, MD 20771
Attn: Warner Miller and Pen-Shu Yeh

K. Sayood, Y.C. Chen, and G. Kipp

Department of Electrical Engineering
University of Nebraska-Lincoln
Lincoln, Nebraska 68588-0511

Period: December 15, 1992 - June 15, 1993

1 Introduction

During this period we continued with the development of simulators for the various HDTV systems proposed to the FCC. The FCC has indicated that it wants the various proposers to collaborate on a single system. Based on all available information this system will look very much like the ADTV system with major contributions only from the DigiCipher system. In this report we describe the results of our simulations of the DigiCipher system. This simulator was tested using test sequences from the MPEG committee. The results are extrapolated to HDTV video sequences.

Once again, some caveats are in order here. The sequences we used for testing the simulator and generating the results are those used for testing the MPEG algorithm. The sequences are of much lower resolution than the HDTV sequences would be, and therefore the extrapolations are not totally accurate. We would expect to get significantly higher compression in terms of bits per pixel with sequences that are of higher resolution. However, the simulator itself is a valid one, and should HDTV sequences become available, they could be used directly with the simulator.

In the following section we give a brief overview of the DigiCipher system. In section 3, we look at some coding results obtained using the simulator. We compare these results to those obtained using the ADTV system. In section 4, we evaluate these results in the context of the CCSDS specifications and make some suggestions as to how the DigiCipher system could be implemented in the NASA network.

Simulations such as the ones reported here can be biased depending on the particular source sequence used. In order to get more complete information about the system one needs to obtain a reasonable set of models which mirror the various kinds of sources encountered during video coding. We have worked on and off on this particular problem. In this report we provide a set of models which can be used to effectively model the various possible scenarios. As this is somewhat tangential to the other work reported here, we have included the results as an appendix.

Also included in the appendix are the various submissions that report work supported under this grant and have appeared in print in the last six months. These are:

- K. Sayood, F. Liu, and J.D. Gibson, "A Joint Source Channel Coder Design," *Proceedings 1993 International Conference on Communication*, Geneva, Switzerland, May 1993, pp. 727-731.

- A.C. Hadenfeldt and K. Sayood, "Compression of Color-Mapped Images," *Proceedings 1993 International Conference on Communication*, Geneva, Switzerland, May 1993, pp. 537-541.

Also, during this period the following were accepted for publication or presentation:

- A.C. Hadenfeldt and K. Sayood, "Compression of Color-Mapped Images," Accepted (with minor revisions) for publication in *IEEE Transactions on Geoscience and Remote Sensing*

- Y.C. Chen and K.Sayood, "The Proposed HDTV Schemes and the NASA Network," Accepted for presentation in *AAIA - Computing in Aerospace 9*

The AAIA paper reports on work done under the current grant. A preliminary draft copy is included in the appendix.

2 DigiCipher System

In this section we briefly describe the DigiCipher system as simulated. The description of the DigiCipher system was obtained from [1]. As our focus is on the compression algorithm, that is the main topic in this section.

2.1 Compression Processor

The compression process can be broken down into six different subprocesses.

1. Chrominance preprocessor

2. Discrete Cosine Transform.

3. Coefficient Quantization (Normalization)

4. Variable Length Encoding

5. Motion Compensation

6. Rate Buffer Control

These are all briefly described below.

2.1.1 Chrominance Preprocessor

Some compression can be obtained directly by reducing the resolution of chrominance information relative to luminance resolution. This has only a slight effect on the perceived image quality. The input signal is first be separated into luminance and chrominance components using the Y, U, V color space representation. The U and V chrominance components are decimated horizontally by a factor of 4, and vertically by a factor of 2. Prior to decimation the pixels are averaged in groups of four horizontally, and groups of two vertically to reduce any aliasing effects. After decimation, the chrominance are multiplexed with the luminance component which bypasses the chrominance preprocessor. All components are then subjected to the same processing.

2.1.2 Discrete Cosine Transform

The DigiCipher system uses the Discrete Cosine Transform (DCT) to decorrelate the input. In this respect it is similar to the ADTV system. Like the ADTV system, the DigiCipher system uses an 8×8 transform.

2.1.3 Coefficient Quantization (Normalization)

The quantization algorithm in the DigiCipher system is a very simple one, consisting of dropping the least significant bits until the‘ number of bits used to represent the coefficient agrees with a pre-arranged value. Because in this way, the amplitude of the coefficient is reduced, the quantization process is also referred to as *normalization*. The number of bits allocated to each coefficient are assigned based on the availability of channel resources. This availability is measured as a function of buffer occupancy. If the buffer is empty each coefficient is assigned nine bits (not counting the sign bit). As the buffer starts to fill up the number of bits assigned to the DCT coefficients is reduced. The reduction is accomplished based on a table shown in Table 2. The occupancy of the buffer is reflected by an index *qlevel* which takes on values between 0 and 9, where 9 denotes an empty buffer, and 0 denotes a full buffer (overflow condition). If the number in the table corresponding to a given coefficient is n then the number of bits used to represent the coefficient is min(9,9-n+*qlevel*). For example, when *qlevel* is 3, all coefficients with n = 9 will get 3 bits, while coefficients with n=8 will get 4 bits and so on.

2.1.4 Huffman (Variable Length) Coding

Once the DCT coefficients are normalized in this manner, they are encoded using a variable length code. The quantized coefficients are arranged as a 64 long vector. The vector is then scanned for non-zero coefficients. When a non-zero coefficient is encountered, its amplitude and the number of zeros preceding it are used as a pointer to a 16 X 16 table which contains the codewords for (runlength, amplitude) pairs from (0,1) to (15,16). That is, this table can code runlengths of upto 15, and amplitudes of upto 16. If the runlength or amplitude exceeds these values a special code is used to tell the decoder not to use the code book to interpret the bits that follow. Instead, the runlength and coefficient amplitude are sent uncoded. The number of bits used to represent the coefficient amplitude is determined by the normalization process described previously. A special end-of-block (EOB) code word is inserted after the last non-zero coefficient.

2.1.5 Motion Compensation

The motion compensation algorithm used in the DigiCipher algorithm is a block based technique. The motion between frames is estimated for each superblock, which has a horizontal dimension equal to 4 DCT blocks, and a vertical dimension equal to two DCT blocks. By all indications, the motion estimation is performed using the luminance block, and the same motion is assumed for the chrominance components. The motion estimation is a simple block matching technique, where a block of pixels in the previous picture which most closely matches the block being encoded is found. A vector (called the motion vector) which describes the displacement of this block relative to the block being encoded is transmitted using 9 bits. The number of bits used to encode the motion vector limits the area of search in the previous picture. The difference image is generated using the difference between these two blocks.

2.1.6 Rate Buffer Control

Because different sections of the image in time and space will have different levels of activity, the coder will generate bits at a variable rate. In order to prevent the video output buffer from overflowing or underflowing, the coding rate needs to be continuously adjusted. To smooth out these variations, a buffer is used with feedback to control the quantization process. As described previously, as the buffer fills up, the number of bits assigned to the coefficients is selectively decreased. This decrease results in a lot of coefficients with an amplitude value of zero which can be efficiently encoded using the runlength procedure described previously, thus reducing the output rate. A lower and upper threshold are defined for the buffer. As long as the occupancy of the buffer remains between the upper and lower thresholds, the quantization strategy is left unchanged. If the buffer level drops below the lower threshold the value of *qlevel* is incremented to increase the quantization resolution and hence the coding rate. If the buffer occupancy increases above the higher threshold, then the value of *qlevel* is decremented to reduce the quantization resolution and hence the coding rate. Fill bits are inserted into the channel in order to prevent underflows during the transmission of very simple images. The document [1] does not describe what steps if any are taken in the case of buffer overflow.

2.2 Data Multiplex Format

Prior to forward error correction each line of data contains 504 information bits. The first four bits in each line are control channel bits, and the next four bits are reserved for data. These are followed by 56 audio bits. In lines 2 through 1050, the next 440 bits are all used for video information. In line

1 bits 503 through 550 are used for framing and synchronization.

Forward error correction is obtained through the use of a 130/154 Reed Solomon coder.

3 Simulation Results

In this section we describe some of the simulation results obtained using the MPEG test sequences. The reconstructed sequences are contained in the accompanying videotape.

Our intent was to generate sequences at similar rates to those described in our previous report for the ADTV scheme. Similar to the ADTV coding scheme, the parameter p is used to control the output rate and length of the rate buffer. We therefore wanted to evaluate the performance of both schemes for similar values of p. However, we found that the DigiCipher system as described in [1] would not perform at those rates. Consider sequence 1 for example. Sequence 1 is coded with $p = 20$ and a buffer window which is [0.3, 0.7] of buffer length. This means that when the buffer fullness is larger than the high threshold (0.7 * buffer length), *qlevel* is decreased by one. On the other hand, *qlevel* is increased by one whenever the the buffer fullness is lower than low threshold. With $p = 20$, the long term mean available capacity is 3.84 Mbits/sec. From Table 1, one can see that the coding rate for this sequence is well above the available rate. In Table 1 we present both the actual rate, and in parenthesis the extrapolated rate for an HDTV transmission. (Note that as the number of active pixels in the ADTV scheme and the DigiCipher scheme are different the conversion ratio is not the same). This means that the results for sequence 1 reflect an unrealistic scenario, where the transmission rate exceeds the available capacity.

The two main differences between the DigiCipher and ADTV compression algorithms are

1. the quantization strategy
2. the frame sequencing

All indications are that this discrepancy in rate is due to the quantization strategy used in the DigiCipher system. As described above, in the DigiCipher system, the quantization level which varies from 0 to 9 is controlled by the fullness of rate buffer. For sequence 1, with the quantization table shown in Table 2, even when *qlevel* is 0, there are still 102 bits (including sign bit) which have to be allocated for each block. Even without any header information, this results in a coding rate of 6.05 Mbits/sec. Under such condition, the buffer is always in the overflow condition and *qlevel* always stays at 0. However, the subjective performance is good because of the unrealistically high coding rate.

In order to decrease the coding rate while maintaining good PSNR performance, we introduced several modifications into the DigiCipher compression algorithm.

Prequantization In the initial modification we introduced a uniform prequantizer with $\Delta = 4$ into the system prior to normalization. The resulting reconstructed sequence is denoted sequence 2. From Table 1, we can see that the prequantization has the effect of reducing the coding rate of Sequence 2 is from 6.48 Mbits/sec to 4.11 Mbits/sec while the average PSNR is down only 0.58 dB. Subjective test shows some slight grain in the sequence which is caused by the pre-quantizer.

It should be noted that, although the value of p is different for sequence 1 and Sequence 2, it is not a determining factor for the coding rate since the *qlevel* is still always 0. Therefore, the only difference between Sequence 1 and 2 is the pre-quantizer. Although the introduction of the pre-quantizer improved the coding performance, it did not relieve the situation of buffer overflow. In order to reduce the rate still further we made a couple of modifications to the quantization algorithm.

Quantization Algorithm Given any variable rate coding approach, it is always possible to come up with an input sequence that will generate a large enough rate for a long enough time to overwhelm the channel resources. Given this fact the rate control algorithm should have an option where the coding rate is set to zero. This can be done in the DigiCipher algorithm if we expand the range of *qlevel* from [0,9] to [-7,9]. We implemented this modification. Along with this we also made the rate control more sensitive by discarding the buffer window, and replacing it with a more sensitive structure. The new algorithm sets the *qlevel* by the degree of buffer fullness. For example, if the buffer is 100% full the value of *qlevel* is set to -7. If the buffer is 94% full the value of *qlevel* is set to -6 and so on. This strategy is similar to that used by the ADTV algorithm

Sequences 3 and 4 are coded with this design which is referred to as $Q2$ in the table and figures. The subjective performance of Sequence 3 is better than Sequence 2 although the average PSNR is lower. The subjective performance of Sequence 4 is acceptable with slight blurriness and graininess, but without any blocking effects. From Table 1 we can see that the coding rate for Sequences 3 and 4 is close to the average long term rate defined by $p = 20$ and 10 respectively. This means that rate buffer is full most of the time although overflow has been avoided. Thus the channel resources are being effectively utilized. From the simulations, it is observed that the *qlevel* values of these two sequences vary between 0 and -7. When *qlevel* is 0, the bit generating rate is higher than the buffer output rate, the buffer fills up, which in turn results in a decreasing *qlevel*. From Figure 1, it is noted that the output rates of the first four sequences are relatively constant. For the first two sequences this is because the value of *qlevel* remains stuck at 0 resulting in a constant bit allocation. For sequence 3, and sequence 4, the full capacity of the channel is being used almost all the time resulting in a constant rate. The frame-by-frame PSNR of the first four sequences is shown in Figure 2. As might be expected the PSNR decreases with decreasing bit rate. However, looking at the results of the subjective tests in Table 1, we notice that while sequence 3 might exhibit poorer PSNR performance than sequence 2, the subjective quality is significantly better. This is in spite of the fact that the bit rate for sequence 3 is less than that for sequence 2. With sequence 4, we are finally in the same range of coding rates as those used for testing the ADTV algorithm in the previous report. However, the subjective quality of sequence 4 is rather poor.

Combination In order to further relieve the saturation of the rate buffer, and improve the quality, both the $Q2$ and pre-quantizer modifications were implemented at the same time. The resulting reconstructed sequences are denoted sequence 5 and sequence 6. The parameter p is set to be 10 and 5 in Sequences 5 and 6 respectively in order to compare the coding performance with the ADTV coding scheme. The coding performance of Sequence 5 is improved both in PSNR and subjective tests compared with Sequence 4. The rates for sequence 4 and sequence 5 are approximately the same. In spite of the modifications, Sequence 6, which is coded with $p = 5$, is very blurred and unacceptable. This seems to indicate that the DigiCipher algorithm is not appropriate for low rates. However, we remind the reader of the caveat mentioned earlier.

Finally, we compare the DigiCipher scheme to the ADTV scheme. Detailed simulations of the ADTV scheme were presented in the previous report. Here we repeat two scenarios. Sequences 7 and 8 are coded with the ADTV technique with p=5 and p=10. Comparing these sequences with sequence 5 and sequence 6 (coded with p=5 and p=10 respectively), we see that the performance using the ADTV algorithm provide better performance, using both objective and subjective performance measure, than the DigiCipher algorithm. It can be reasonably concluded that the ADTV compression scheme is more efficient than the DigiCipher algorithm (at least at lower rates).

4 DigiCipher Transmission on the CCSDS Network

Our recommendations for the DigiCipher system are identical to the recommendations for the ADTV system. In our previous report we recommended the dedication of a Virtual Channel for transmission of ADTV coded information. This was in spite of the fact that ADTV data tends to be somewhat bursty. In the case of the DigiCipher system, our simulations indicate a very flat rate profile which argues even more strongly for the dedication of a virtual channel. This would then be transmitted as isochronous data by the Space Link Subnet (SLS) using the Bitstream service.

As argued in the previous report, the delay constraints on the transmission preclude the use of the Space Link ARQ procedure, while the delay constraint coupled with the high rate argue against the use of the Insert and Multiplexing procedures. This leaves the Bitstream procedure as the only viable candidate for HDTV transmission. This conclusion coincides with that of the CCSDS Red Book *Audio, Video, and Still-Image Communication Services*

As far as the grade of service is concerned, one could use the error protection service provided by the DigiCipher algorithm with grade 3 service. Forward error correction in the DigiCipher system is provided by a Reed-Solomon encoder. Some error detection is also provided by the use of End-of-Block codes. The EOB codes provide a means for recovery as well as error detection. Recovery is further aided by the use of a 24 bit frame sync and a 15 bit pointer to the next macro block.

Another option would be to discard any error protection from the DigiCipher signal and use grade-2 service. It is not clear at present which would be the preferable option. During the next period we will be examining different noisy channel scenarios. We plan to address this question as well as similar questions with respect to the ADTV algorithm during this period.

References

[1] R. Rast, General Instruments Corporation, "General Instruments DigiCipher HDTV System," *All Digital High Definition Television Symposium,* Columbia Univ. Center for Telecommunication Research, April 1991.

	Mean Rate (Mbits/sec)	STDR	PAR	Average PSNR	STDP	Subjective Test
Sequence 1	6.48 (103.68)	0.89	2.63	37.91	1.99	very good
Sequence 2	4.11 (65.76)	0.17	1.30	37.33	1.34	slightly grainy, but O.K.
Sequence 3	3.93 (62.88)	1.09	4.33	35.70	1.87	very good
Sequence 4	1.96 (31.36)	0.51	4.05	33.76	0.98	blurred, grainy, no blocky effect
Sequence 5	1.94 (31.04)	0.30	2.73	34.36	1.15	grainy, blurred, but O.K.
Sequence 6	0.98 (15.68)	0.33	5.02	32.53	0.79	very blurred, unacceptable
Sequence 7	1.91 (30.56)	0.44	1.77	41.28	1.21	very good
Sequence 8	1.04 (16.64)	0.61	3.24	38.62	1.72	slightly grainy

(): coding rate interpolated with DigiCipher format

STDR: Standard deviation of coding rate(Mbits/sec)

PAR: Peak to average ratio

STDP: Standard deviation of PSNR(dB)

Sequence 1: DigiCipher, p=20, Q1, QS=1

Sequence 2: DigiCipher, p=10, Q1, QS=4

Sequence 3: DigiCipher, p=20, Q2, QS=1

Sequence 4: DigiCipher, p=10, Q2, QS=1

Sequence 5: DigiCipher, p=10, Q2, QS=4

Sequence 6: DigiCipher, p=5, Q2, QS=4

Sequence 7: ADTV, p=10, QS=4 for INTRA frame

Sequence 8: ADTV, p=5, QS=4 for INTRA frame

p: Bandwidth = px192 kbits/sec

Susie video pixels: (Luma) 352Hx240V, (Chroma) 176Hx120V, 30 frames/sec

DigiCipher video pixels: (Luma) 1408Hx960V, (Chroma) 350Hx480V

Table 1 Performance of coding rate and PSNR using DigiCipher and ADTV coding schemes (Susie sequence 150 frames).

2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	9
4	5	6	7	8	9	9	9
5	6	7	8	9	9	9	9
6	7	8	9	9	9	9	9
7	8	9	9	9	9	9	9
8	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9

Table 2 Bit allocation table for 8x8 DCT coefficients.

Figure 1. Performance of coding rate
for DigiCipher system.

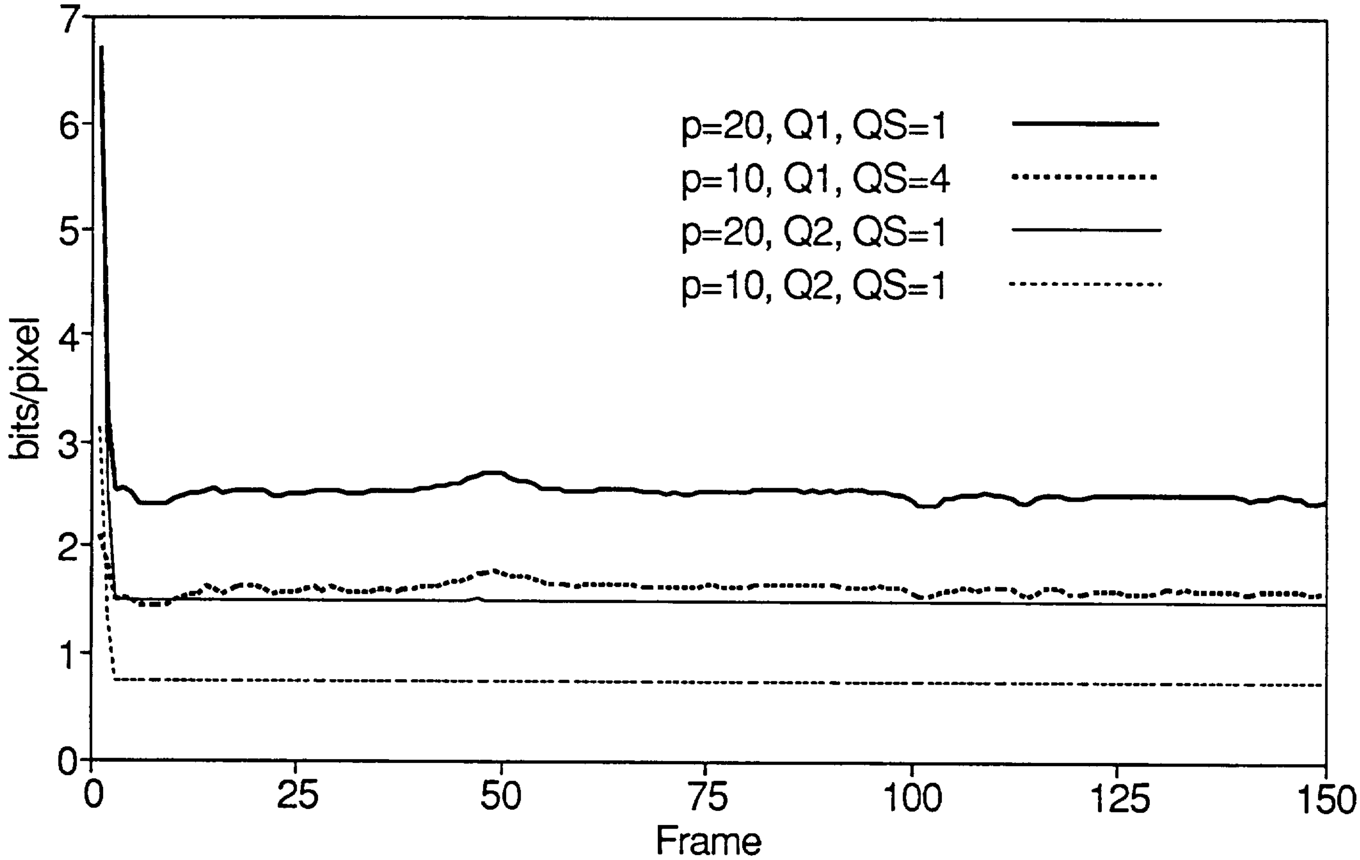


Figure 2. Performance of PSNR for DigiCipher system.

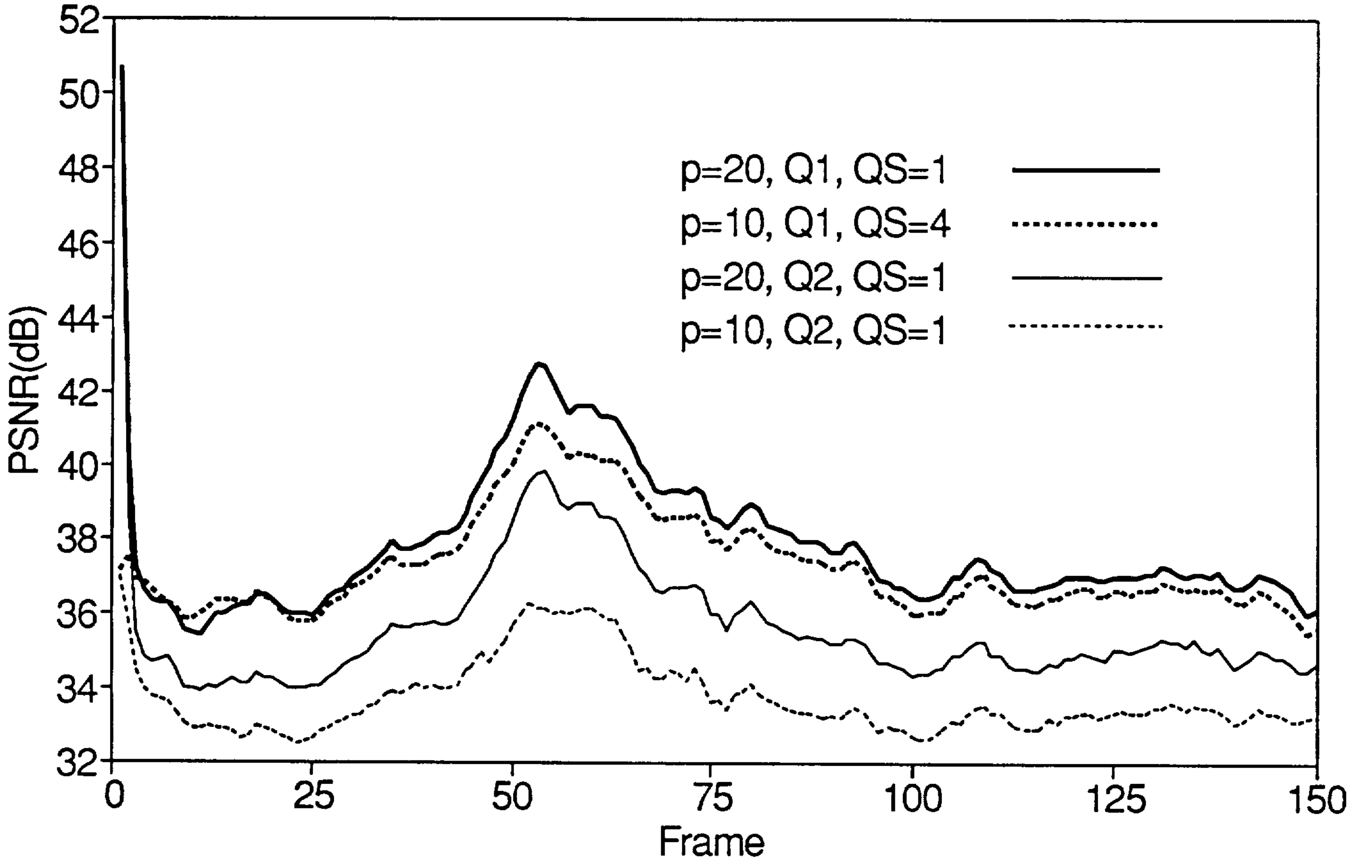


Figure 3. Performance of coding rate for DigiCipher and ADTV system.

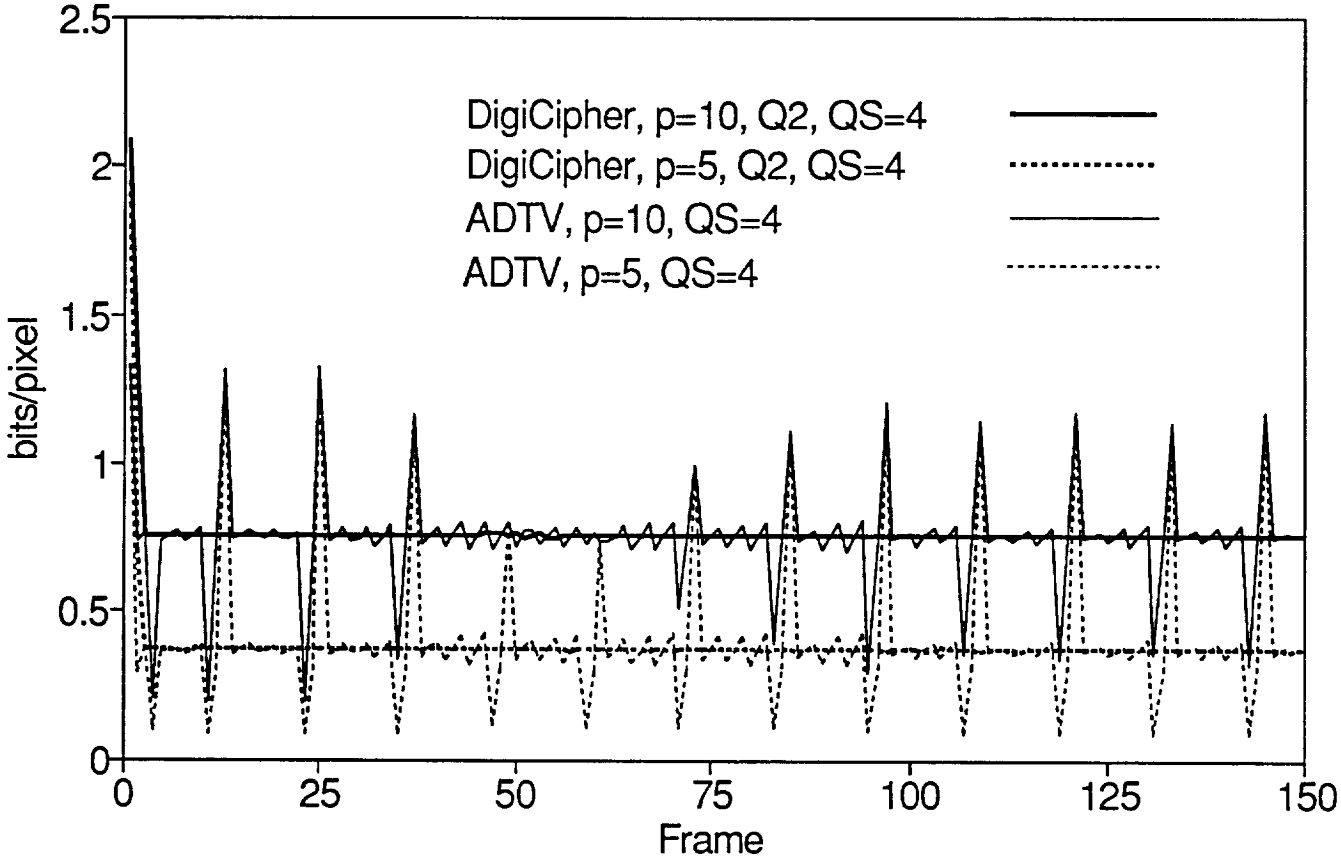
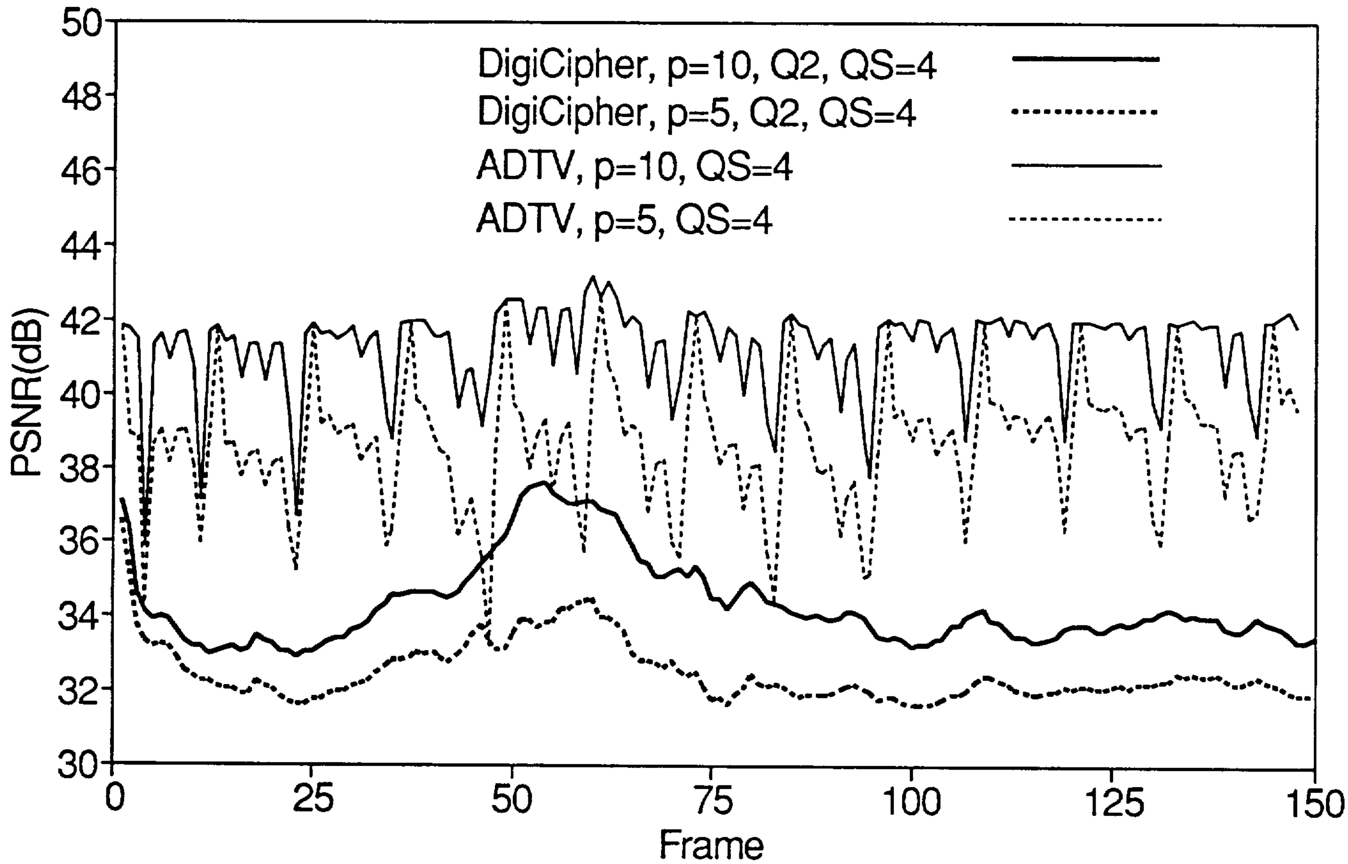


Figure 4. Performance of PSNR for DigiCipher and ADTV system.



Appendices

Video Source Modelling

Variable bit rate coding scheme will be implemented in ATM networks in order to obtain flexibility and efficiency. This is important because the output bit rate of a video source depends on the specific scene content and coding scheme used. Also, different types of video sources will have different statistical characteristics. Thus it would be inefficient to use constant bit rate coding schemes and peak bandwidth allocation. We are interested in transmitting the coded video information across ATM networks efficiently. Performance simulation is very important when designing a coding scheme which will hopefully best fit into the future ATM environment. Efficient and accurate simulation depends on accurate modelling. Unfortunately, modelling of a video source is more difficult when compared to speech source since video is a highly complex source. When developing a model, one has to decide on the degree of complexity and sophistication required of the model. Too much attention to detail can lead to models that are mathematically or computationally intractable. Our objective in modelling the video sequences is relatively modest. We would like to closely match the second order statistics of the data. This model can then be used both in simulation as well as possibly for prediction of rate fluctuations.

Various video source models have been proposed. For simulation purpose, the source are modelled with a first order AR process by Nomura *et al.* [1] and Maglaris *et al.* [2].

A Markov chain model was proposed by Heyman *et al.* [3]. For queueing analysis, Maglaris *et al.* [2] and Sen *et al.* [4] model the source as a birth-death Markov process. A more elaborate model which combine first-order AR process and a three state Markov chain was presented by Ramamurthy *et al.* [5]. Melamed *et al.* [6] model a VBR video source by exploiting bit-rate histogram and autocorrelation fuction. In the following sections, we introduce some models to simulate the variable bit rate of a video source. We will also validate these models by performing the goodness-of-fit tests.

1 Video Source Sequences

As mentioned above, different type of video sources generate sequences with different statistical characteristics, and different bit rates. In this chapter, two sequences of different nature will be used to explore the capability of models. The first sequence is *Susie* sequence, a MPEG standard, under different coding modes. Figure 1 shows the bit-rate (averaged for each frame) for all 600 frames (20 seconds) using MBCPT and various thresholds. The average value μ over all 600 frames and the standard deviation σ were found to be $\mu = 0.82$ bits/pixel and $\sigma = 0.1958$ bits/pixel. The maximum value of the bit-rate is 1.39 bits/pixel and the minimum value of the bit-rate is 0.33 bits/pixel.

In order to explore the long term correlation, we interpolate *Susie* and *Football* in the second sequence which lasts 900 frames. This sequence features four scene cuts which occur at frame number 150, 300, 600 and 750, as shown in Figure 2. The mean value μ and the standard deviation σ are 1.29 and 0.7081 bits/pixel respectively. The maximum value of the bit-rate is 3.15 bits/pixel and the minimum value of the bit-rate is 0.33

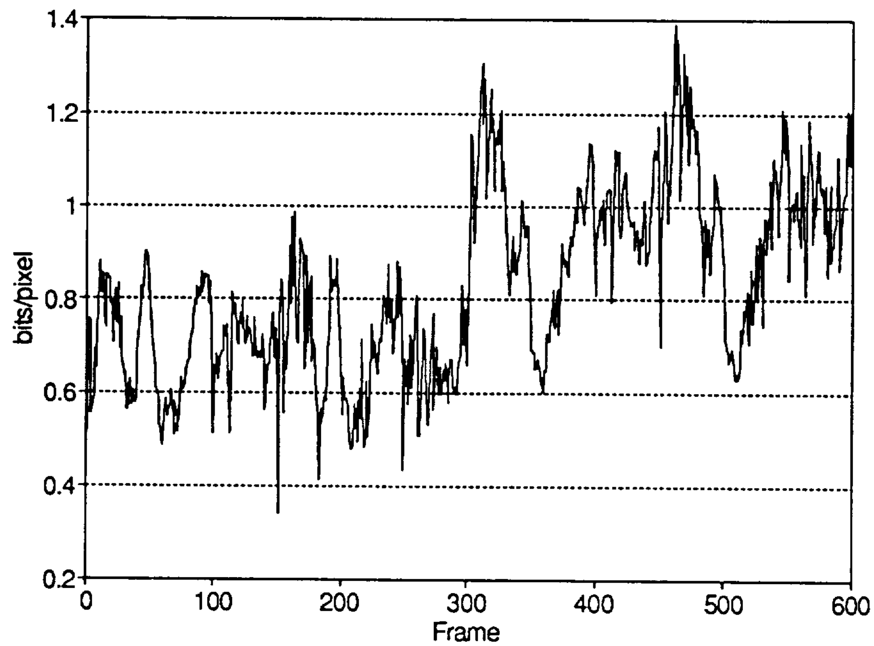


Figure 1 Coding bit-rate of Sequence 1.

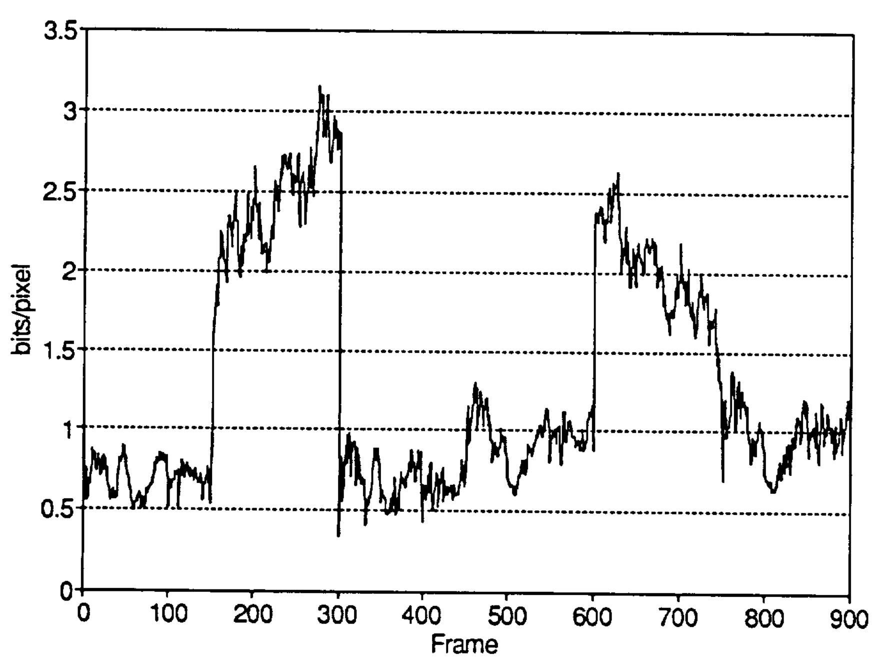


Figure 2 Coding bit-rate of Sequence 2.

bits/pixel. For the rest of this chapter, these two sequences were referred to as Sequence 1 and 2.

2 Models for Homogeneous Sequence

From the recorded data in Figure 1, there are not too many discontinuities and very steep changes were observed. Therefore we concluded that it is a smooth sequence and hoped that the following models can accurately describe the behavior of bit-rate output.

2.1 Continuous State Autoregressive Markov Model

An autoregreesive process of order p (or AR(p)) can be expressed as

$$X_n = \sum_{i=1}^{p} a_i X_{n-i} + Z_n \tag{1}$$

where Z_n is a sequence of white noise. Our aim is to find the coefficients vector $\mathbf{A} = (a_1,...,a_p)$ and the white noise variance σ^2 based on the bit-rate $\lambda_1,...,\lambda_n$. For stationary AR process, the coefficients of the AR model is related to the autocorrelation coefficients by the *Yule-Walker* equations [7]. Yule-Walker estimator is used to do the preliminary estimation. The preliminary estimated model is then optimized using *maximum likelihood* estimator. For AR(1), we obtained

$$X_n = 0.829 + 0.907X_{n-1} + Z_n, \qquad Z_n \sim WN(0,0.007042) \tag{2}$$

When an AR model is fitted to a given series, an essential part of the procedure is to examine the residuals, which should, if the model is satisfactory, have the appearance of a realization of white noise. If the autocorrelations and partial autocorrelations of the residuals suggest that they come from some other identifiable process, then some other models are recommended. To test for independence in the residuals of this AR(1) model, some randomness test are applied, which includes *Portmanteau* test, *Turning-Point* test, *Difference-Sign* test and *Rank* test. Test results show that the residuals from this AR(1) model is not white. A natural response for decorrelating the residuals is to increase the order of AR model. Figure 3 shows that the autocorrelation function of AR(4) model has a better match for the empirical autocorrelation. Yule-Walker and Maximum-Likelihood estimator provide the following AR(4) model.

$$\begin{aligned} X_n = 0.829 &+ 0.770X_{n-1} - 0.021X_{n-2} + 0.093X_{n-3} \\ &+ 0.096X_{n-4} + Z_n, \qquad Z_n \sim WN(0,0.006698) \end{aligned} \tag{3}$$

Based on the satisfactions of above tests, it is concluded that AR(4) was a suitable model for Sequence 1.

2.2 Discrete Time Markov Chain Model

A Markov process with a discrete state space is referred to as a Markov chain. To set up the model, the bit-rate λ_n is quantized into discrete levels using a uniform quantizer with stepsize 0.05 bits/pixel. Each quantization level is a state of Markov chain model. In the simulation, the number of states is chosen as 13 in order to cover the range of bit-rate values. With Sequence 1, one-step transition probability r_{ij} is estimated in the usual way:

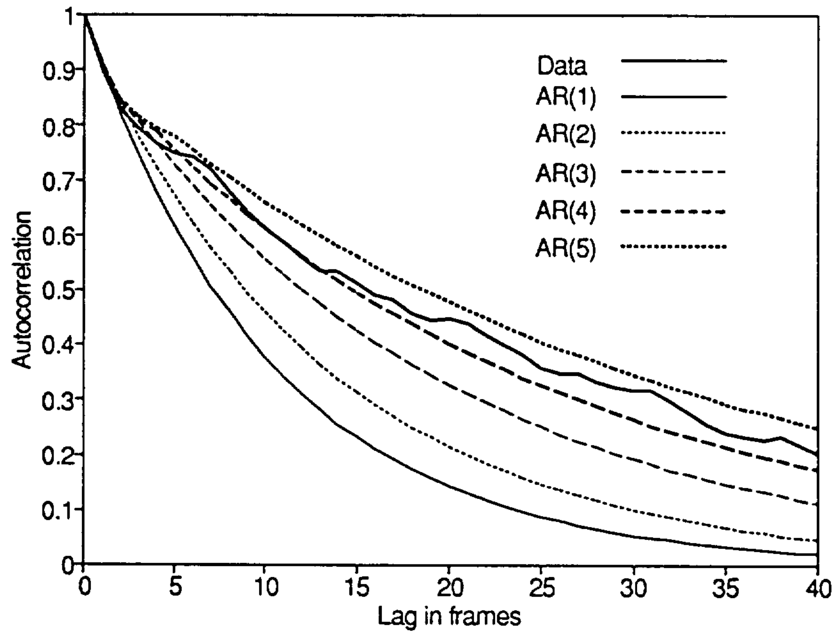


Figure 3 Autocorrelation functions of Sequence 1 and AR(1)-AR(5).

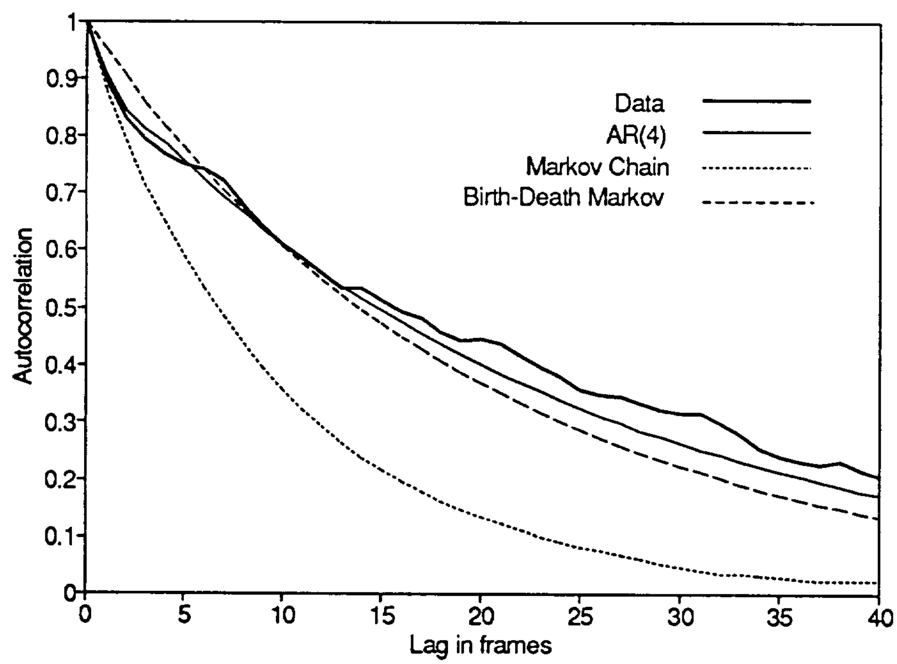


Figure 4 Autocorrelation function of several models (Sequence 1).

$$r_{ij} = \frac{number\ of\ transitions\ i\ to\ j}{number\ of\ transitions\ out\ of\ i} \tag{4}$$

when the denominator is greater than zero. When the denominator is zero, p_{i1} is set to be 1. Since there is no transition out of state i, this assignment will not affect the stationary distribution. The autocorrelation function of the Markov chain model, as shown in Figure 4, does not match that of Sequence 1 well. Accuracy can be improved by refining the quantization stepsize.

2.3 Discrete Time Birth-Death Markov Model

In this section, the possibility of modelling "smooth" sequence with a birth-death Markov model, which has been proposed in [2], is investigated with Sequence 1. Unlike Markov chain model, birth-death markov model allows only transitions between neighboring states which reflects the fact that there won't be any steep changes in the process. According to the nature of bit-rate sequence, it is assumed that there exists a tendency of the bit-rate toward higher levels to decrease at high levels, and inversely, the tendency of the bit-rate toward lower levels to increase at higher levels. This is a reasonable assumption and will result in a bell shaped stationary distribution of the state. In the model, bit-rate is

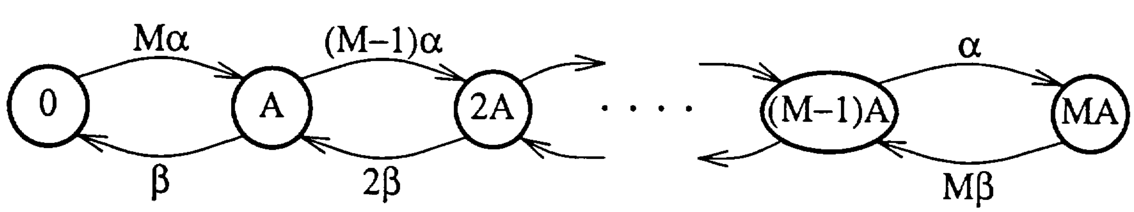


Figure 5 State-transition-rate diagram for birth-death Markov model.

quantized with uniform quantization step A bits/pixel, and $M + 1$ possible levels, $(0, A, \ldots, MA)$. The transition rates $r_{i,j}$ from state iA to jA are given by

$$\begin{aligned} r_{i,i+1} &= (M-i)\alpha & i < M \\ r_{i,i-1} &= i\beta & i > 0 \\ r_{i,i} &= 0 & \\ r_{i,j} &= 0 & |i-j| > 1 \end{aligned} \tag{5}$$

Figure 5 shows such a birth-death process. It is easy to show [8] that λ_n at steady state has a binomial distribution with mean $E(\lambda)$, variance $C(0)$ and exponential autocovariance $C(\tau)$

$$\begin{aligned} P\{\lambda_n = kA\} &= \binom{M}{k} p^{k}(1-p)^{M-k} \\ p &= \frac{\alpha}{\alpha+\beta} \\ E(\lambda) &= MAp \\ C(0) &= MA^2 p(1-p) \\ C(\tau) &= C(0)e^{-(\alpha+\beta)\tau} \end{aligned} \tag{6}$$

The parameters of this model A, α, β can be estimated by matching above equations with the measured values and take M as a parameter. Note that the autocorrelation function is fitted by an exponential equation of the form $C(\tau)/C(0) = e^{-a\tau}$ by sampling at 30 frames/s, where τ is an unit in second and a is the best matching coefficient. From Figure 6, we take $a = 0.05/(1/30) = 1.5$. In this binomial model, the rate λ_n can be thought of as the aggregate rate from M independent minisources, each alternating between transmitting 0 (the OFF state) and A bits/pixel (the ON state) as shown in Figure 7.

We assume a continuous state queue which is fed by M independent minisources, each sends λ units of flow per time unit when it is on. The queue empties with fixed rate μ units of flow per time unit, also shown in Figure 7. A two dimensional state $\{q(t), k(t)\}$

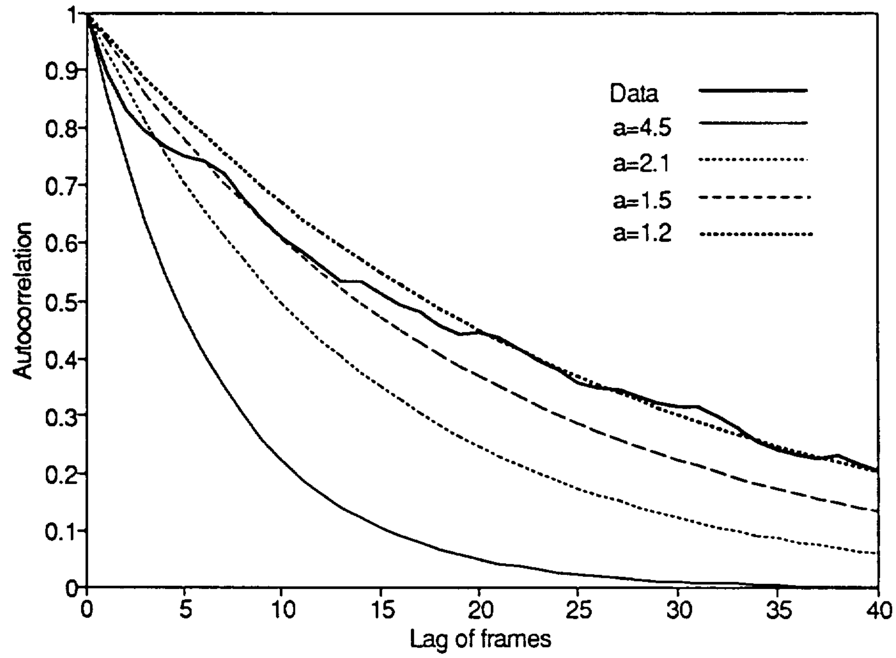


Figure 6 Exponential fits ($e^{-a\tau}$, τ = lag/30) of autocorrelation function (Sequence 1).

is built to completely describe the queueing system. The first component represents the length of queue at time t, while the second component shows how many minisources are on at that instant. Figure 8 shows the state-transition-rate diagram and Figure 9 shows the instantaneous transition rate matrix Q, for M equals 3. Q can be represented in another form with defined A, B, C. We define $\pi = [\pi_0, \pi_1, \pi_2, \ldots]$ as the limiting state probability where $\pi_i = [\pi_{i0}, \pi_{i1}, \ldots, \pi_{iM}]$, π_{ij} is the limiting probability of the state with queue length i and there are j minisources on. For an ergodic Markov chain, we can express equilibrium solution, through *Chapman-Kolmogorov* equation, in matrix form as

$$\pi Q = 0 \tag{7}$$

This last equation coupled with the probability conservation relation, namely

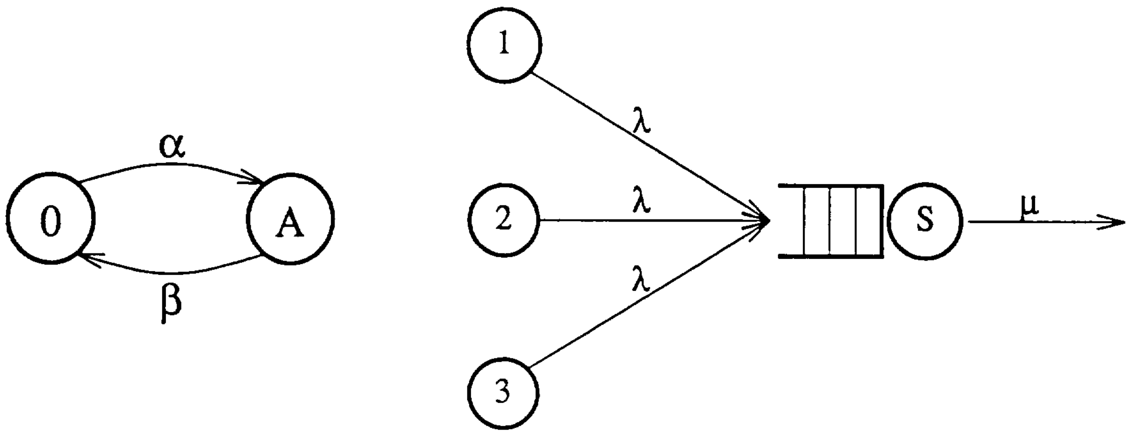


Figure 7 Queueing system for minisource model (M = 3).

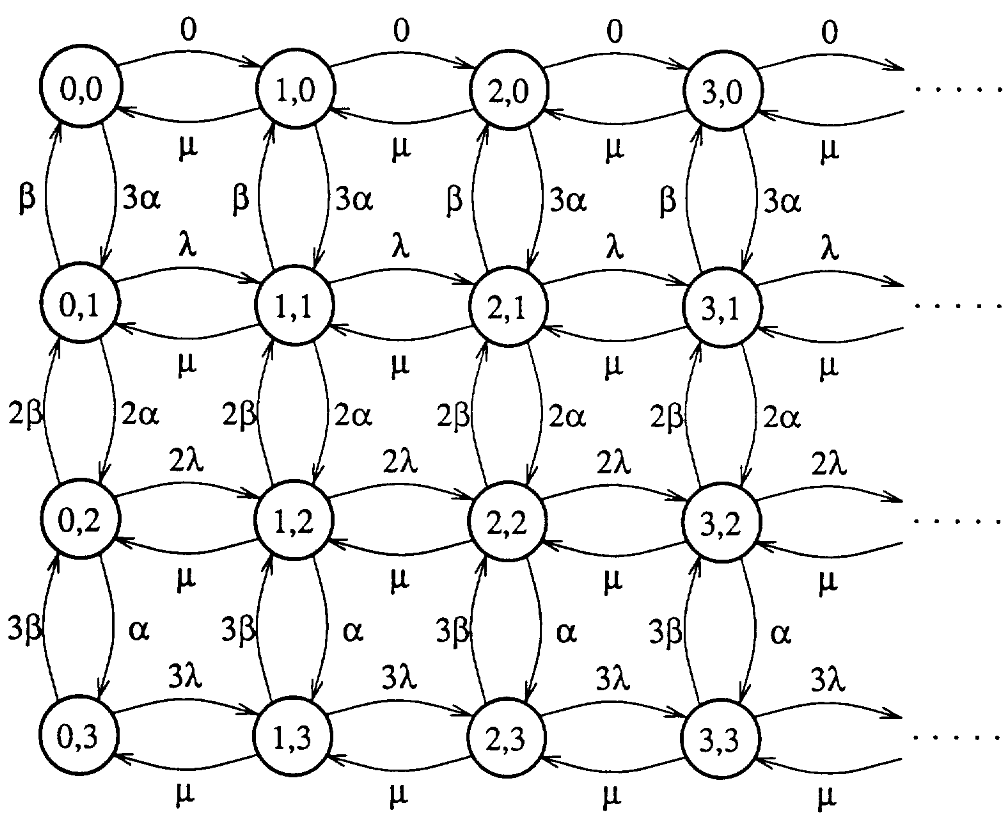


Figure 8 State-transition-rate diagram for minisource (M = 3).

$$
\begin{array}{c|cccc|cccc|cccc}
 & 00 & 01 & 02 & 03 & 10 & 11 & 12 & 13 & 20 & 21 & 22 & 23 \\
\hline
00 & -3\alpha & 3\alpha & & & 0 & & & & & & & \\
01 & \beta & -(\lambda+2\alpha+\beta) & 2\alpha & & & \lambda & & & & \cdots & & \\
02 & & 2\beta & -(2\lambda+\alpha+2\beta) & \alpha & & & 2\lambda & & & & & \\
03 & & & 3\beta & -(3\lambda+3\beta) & & & & 3\lambda & & & & \\
\hline
10 & \mu & & & & -(3\alpha+\mu) & 3\alpha & & & 0 & & & \\
11 & & \mu & & & \beta & -(2\alpha+\beta+\lambda+\mu) & 2\alpha & & & \lambda & & \\
12 & & & \mu & & & 2\beta & -(\alpha+2\beta+2\lambda+\mu) & \alpha & & & 2\lambda & \\
13 & & & & \mu & & & 3\beta & -(3\beta+3\lambda+\mu) & & & & 3\lambda \\
\hline
 & & & \vdots & & & & \vdots & & \ddots & & &
\end{array}
$$

$$
A = \begin{pmatrix} -3\alpha & 3\alpha & & \\ \beta & -(\lambda+2\alpha+\beta) & 2\alpha & \\ & 2\beta & -(2\lambda+\alpha+2\beta) & \alpha \\ & & 3\beta & -(3\lambda+3\beta) \end{pmatrix} \qquad
B = \begin{pmatrix} 0 & & & \\ & \lambda & & \\ & & 2\lambda & \\ & & & 3\lambda \end{pmatrix}
$$

$$
C = \begin{pmatrix} \mu & & & \\ & \mu & & \\ & & \mu & \\ & & & \mu \end{pmatrix} \qquad
Q = \begin{pmatrix} A & B & 0 & \cdots \\ C & A\text{-}C & B & \cdots \\ 0 & C & A\text{-}C & \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}
$$

Figure 9 Instantaneous transition rate matrix of minisource model.

$$\sum_i \pi_i = 1 \tag{8}$$

uniquely gives us the limiting state probabilities. With Q expressed as in Figure 9, we have

$$
\begin{aligned}
\pi_0 A + \pi_1 C &= 0 \\
\pi_{i-1} B + \pi_i (A - C) + \pi_{i+1} C &= 0 \qquad \textit{for } i \geq 1
\end{aligned}
\tag{9}
$$

Many methods are available for solving this set of equations, for example the method of z-transforms. Here, we just use a simple computational approach. For a stationary process, we assume

$$
\pi_{i+1} = \pi_i R \qquad \textit{for } i \geq 1 \tag{10}
$$

Substitute Eq. (10) into the second equation of Eq. (9), we obtain

$$
B + R(A - C) + R^2 C = 0 \tag{11}
$$

Rearrange the last equation, we have

$$
R = -(R^2 C + B)(A - C)^{-1} \tag{12}
$$

We can solve R simply using computer iteration with initialization of R to be null matrix. Applying Eq. (8) and the first equation of Eq. (9) we obtain

$$
\begin{aligned}
\pi_1(-CA^{-1} + 1 + R + R^2 + \cdots) &= F \\
\pi_1(-CA^{-1} + \frac{1}{1-R}) &= F \\
\pi_1 &= F(-CA^{-1} + \frac{1}{1-R})^{-1}
\end{aligned}
\tag{13}
$$

where F is the vector of binomial probability distribution from the first equation of Eq. (6). Since we found π_1, π follows using Eq. (9) and (10). We define $P = [p_0, p_1, p_2, ...]$ as equilibrium probability distribution of queue length. It is easy to obtain P by

$$p_i = \sum_{j=0}^{M} \pi_{ij} \tag{14}$$

Analysis results will be presented and discussed in section 4.2.

3 Models for Scene-Cut Sequence

In this section, three models are used to simulate Sequence 2. Autoregressive and Markov chain models are remained. Since the nature of birth-death model does not meet the appearance of sequence 2, It is replaced with another model, namely Hidden Markov Model.

3.1 Another Autoregreesive and Markov Chain Model

Repeating the same procedure as in Section 2.1 and 2.2, we obtain an autoregressive model with order 2 as

$$X_n = 1.291 + 0.814X_{n-1} + 0.169X_{n-2} + Z_n, \quad Z_n \sim WN(0,0.018910) \tag{15}$$

It seems an AR(2) model is good enough to describe Sequence 2, either in autocorrelation function or in randomness of residuals. Markov chain model is the same as in section 2.2 except a coarser quantizer and a larger number of states are used to cover the wide range of Sequence 2. The quantization stepsize is 0.1 bits/pixel and there are 29 states in our simulation. Figure 10 shows that the autocorrelation functions of these two models are reasonably close, atleast up to 40 frames, to that of Sequence 2.

3.2 Hidden Markov Model (HMM)

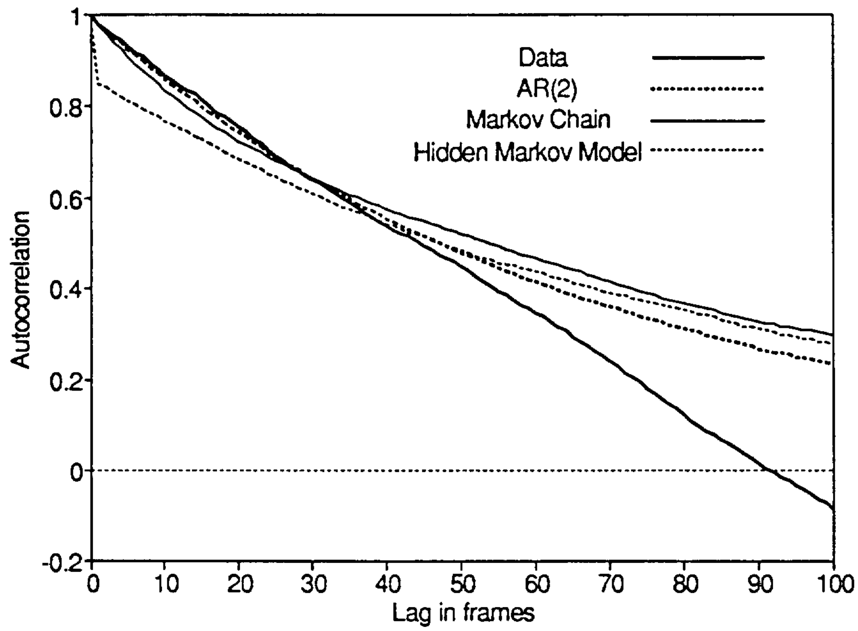


Figure 10 Autocorrelation function of several models (Sequence 2).

The hidden Markov model (HMM) is a doubly embedded stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observations [9]. An HMM is characterized by the following:

1. N, the number of states in the model.
2. M, the number of distinct observation symbols per state.
3. $A = \{a_{ij}\}$, the state transition probability.
4. $B = \{b_j(k)\}$, the observation symbol probability.
5. $\pi = \{\pi_i\}$, the initial state distribution.

Building a HMM follows the following three procedures:

1. Given the observation sequence $O = O_1 O_2 \cdots O_T$ and a model $\lambda = (A, B, \pi)$, compute $P(O|\lambda)$, the probability of the observation sequence, given the model.
2. Given the observation sequence O and the model λ, choose a corresponding state sequence $Q = q_1 q_2 \cdots q_T$, which is optimal in some meaningful sense (i.e., best "explains" the observations).
3. Adjust the model parameters $\lambda = (A, B, \pi)$ to optimize $P(O|\lambda)$.

Since Sequence 2 is generated from two different types of video, attaching the frame's content to the state of HMM seems to be a natural choice. Two states are defined in HMM model, namely low-motion and high-motion state, therefore N equals 2. We hope this choice can accurately reflect the bit rate distribution in different motion sequences. The observation symbols are the quantized bit-rate output of Sequence 2. Number of distinct observation symbols M is set to be 29 using the same quantizer as in Markov chain model. There are several possible ways of initializing the algorithm used for developing the HMM. These depend on the selection of matrix A, B, π. As suggested in [9], an uniform estimates for π and A parameters are adopted. As for B matrix, weighted parameters are assigned in order to obtain the global maximum of the likelihood function in the algorithm.

4 Goodness-of-Fit Tests

In this section some statistics, simulation and queueing analysis results are used to test goodness-of-fit for the models introduced above.

4.1 Statistics

For each statistical model we generate 10 realizations of 5,000 frames. The 10 realizations are treated as independent and identically distributed samples. Table 1 shows mean, variance along with 95% confidence intervals which is expressed as [10]

$$P[\bar{x} - t_{0.025}(\frac{s}{\sqrt{n}}) \le x \le \bar{x} + t_{0.025}(\frac{s}{\sqrt{n}})] = 0.95 \qquad \textbf{(16)}$$

where s is sample's standard deviation, n is the number of samples which is 10. The value $t_{0.025}$ equals 2.262 corresponding to 9 degrees of freedom for *Student's t* distribution. Note that it is possible for AR(2) model to generate negative values which is not allowed in our application. Since this situation does not occur often, negative values are simply replaced with zero and the statistics does not change drastically. Table 1 reveals that all confidence intervals include the values from the real data except the one of variance when

	Sequence 1		Sequence 2	
	Mean	Variance	Mean	Variance
Data	0.8296	0.0383	1.2905	0.5014
AR	0.8292±0.0134	0.0392±0.0027	1.3251±0.0657	0.4303±0.0444
MC	0.8274±0.0090	0.0385±0.0012	1.3035±0.0613	0.4763±0.0324
HMM	-------	-------	1.2825±0.0621	0.4869±0.0326

Table 1 Statistics with 95% confidence interval.

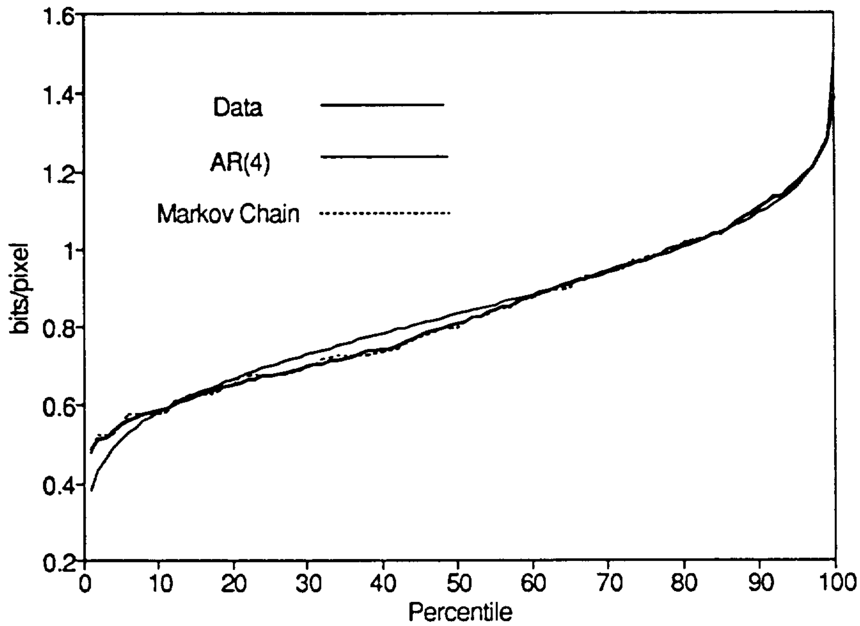


Figure 11 Percentile plot of several models (Sequence 1).

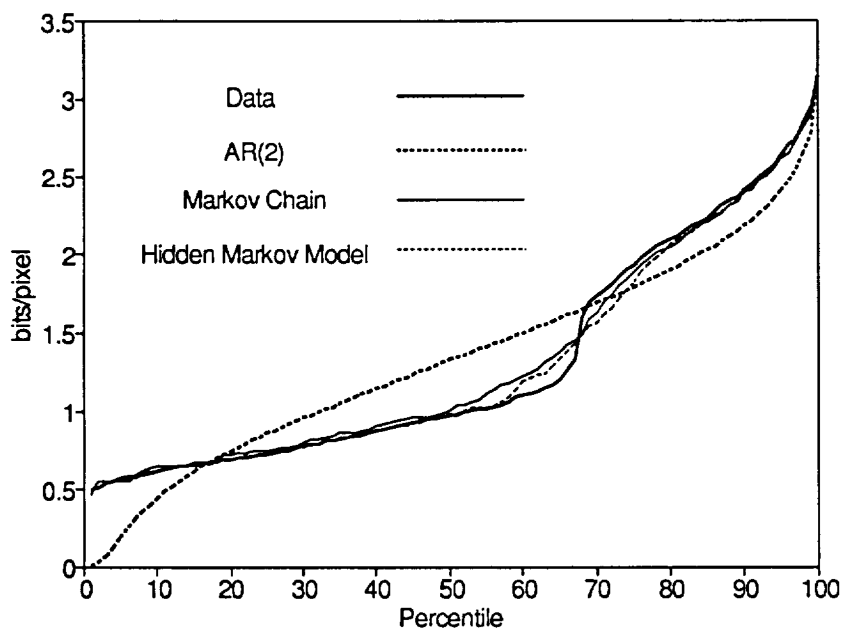


Figure 12 Percentile plot of several models (Sequence 2).

applying AR(2) model in Sequence 2. Statistical test of hypothesis may suggest rejection of AR(2) model at this point. Another powerful tool to test goodness-of-fit is *percentile* plot which draws the percentile of distribution. As shown in Figure 11, Markov chain model shows a very good fit for the data from Sequence 1. AR(4) model fits the data reasonably well except some region with small deviation. Figure 12 again shows that AR(2) model is not a good model for sequence with clumps of large value, like Sequence 2. On the other hand, Markov chain and HMM satisfactorily follow the big jump of data.

4.2 Cell Loss Performance for Homogeneous Sequence

Cell loss is one of most concerned problem when transmitting information through ATM networks. Therefore, besides fitness of statistics, queueing performance is another important consideration in video source modelling [11]. We design two queueing systems to perform the evaluation. The first one is a finite buffer queue where buffer space corresponds to transmission delay. Every arriving cell is discarded when the buffer is full. The second one is a queue with infinite buffer space. The reason this queueing model is proposed is to compare the performance of minisource model with other models. All coding bits from a single frame are assembled into cells with length equals 384 bits, according to CCITT specification. All the cells from the same frame are equally spaced in a frame duration, 1/30 second. The server serves with a determinant rate which equals average video rate divided by a utilization factor.

***Case* 1**: Queueing model with finite buffer

	Cell Loss Probability for Various Maximum Delay Allowed				
	5ms	10ms	20ms	50ms	100ms
Data	0.0195	0.0188	0.0176	0.0149	0.0121
AR(4)	0.0185±0.0032	0.0178±0.0032	0.0168±0.0032	0.0151±0.0033	0.0131±0.0033
MC	0.0204±0.0022	0.0196±0.0021	0.0183±0.0021	0.0156±0.0020	0.0125±0.0019

Table 2 Cell loss probability of several models (Sequence 1, Case 1).

Figure 13 shows cell loss probability of Sequence 1, AR(4) and Markov chain model from simulation with utilization factor ρ equals 0.8. From the transmission rate calculated, 1 ms delay is approximately equivalent to the transmission time of one cell. Therefore, the cell loss probability corresponding to, say 10 ms delay, is the simulation result with a 10 cell buffer. For AR(4) and Markov chain model, simulations are run with 10 realizations each. The values shown in Figure 13 are the average of 10 simulations. The difference between the performances of data and models seems small compared with the total cells generated. However, we would like to validate the fitness of these models by checking the test of hypotheses again using t statistics. Table 2 shows that AR(4) and Markov chain model easily pass the test.

***Case* 2**: Queueing model with infinite buffer

This queueing model is similar to the previous one but with an infinite buffer space. Therefore all arriving cells are put in the buffer without loss. We collect time average

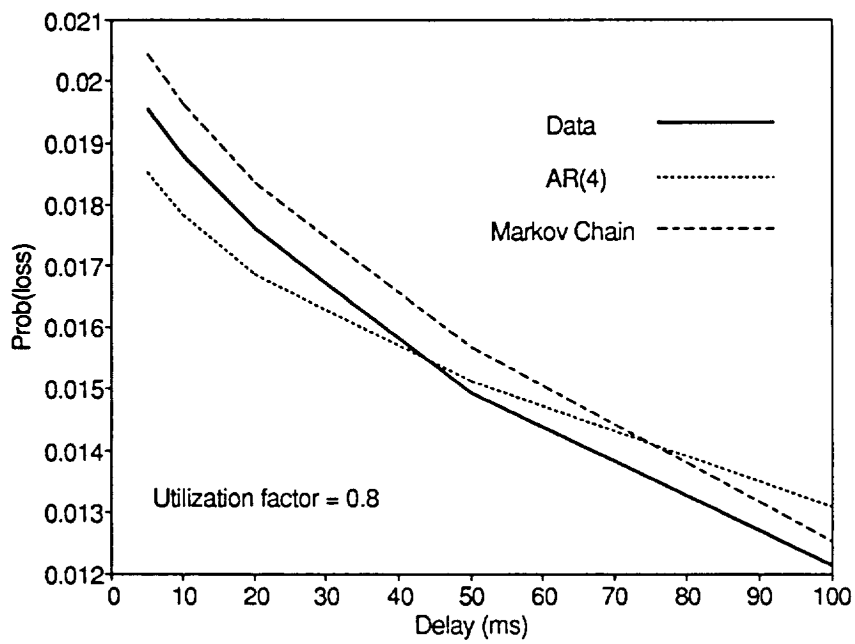


Figure 13 Cell loss probability of several models (Sequence 1, Case 1).

statistics to find the distribution of buffer length. Probability of cell loss in this case is equivalent to the probability that queue length exceeds some certain value k. It means that the arriving cell who finds there are more than k cells waiting in the queue is deemed to be lost because it will be too late for video reconstruction in the receiver end. As in the previous case, simulations have been run using Sequence 1 and 10 realizations of each AR(4) and Markov chain model. Comparison between simulation results and queueing analysis result of minisource model will be made.

As described in Section 2.3, we were able to find the equilibrium distribution of queue length P in minisource model. The parameters of the model M, A, α, β are obtained by matching Eq. (6) with measured values. With M as a parameter, we have

$$
\begin{aligned}
\beta &= a \,/\, (1+\frac{E^2(\lambda)}{M\cdot C(0)}) \\
&= 1.5 \,/\, (1+\frac{0.6883}{M\cdot 0.0383}) \\
\alpha &= a - \beta = 1.5 - \beta \\
A &= \frac{C(0)}{E(\lambda)} + \frac{E(\lambda)}{M} = 0.0462 + \frac{0.8296}{M}
\end{aligned}
\tag{17}
$$

In order to cover the value of maximum bit-rate, M is chosen to be 13. In this case, α and β are 0.8699 and 0.6301 per second respectively and A is 0.11 bits/pixel. We define λ as the cell generating rate per second per minisource and μ as cell transmission rate per second. Utilization factor is set to be 0.8.

Figure 14 shows the simulation and analysis results. Note that minisource model overestimates cell loss probability in the short delay region and underestimates that in the long delay region. The overestimation is caused by the fact that Sequence 1 does not contain low bit-rate which consequently make α large. Large value of α means that if a minisource is on it tends to stay on. In the case of short buffer, this kind of behavior causes cell loss. A long buffer can absorb this effect and reflect the "smooth" characteristic of a birth-death process. However, it is this "smooth" property causes the underestimation in the long delay area. From the observation of Sequence 1, there are some amount of transitions between states which are not neighboring. Nevertheless, minisource model is still a power analytic tool especially when investigating the behavior of statistical multiplexing.

4.3 Cell Loss Performance for Scene-Cut Sequence

From Figure 2 we observed the durations which contain low and high bit-rate value

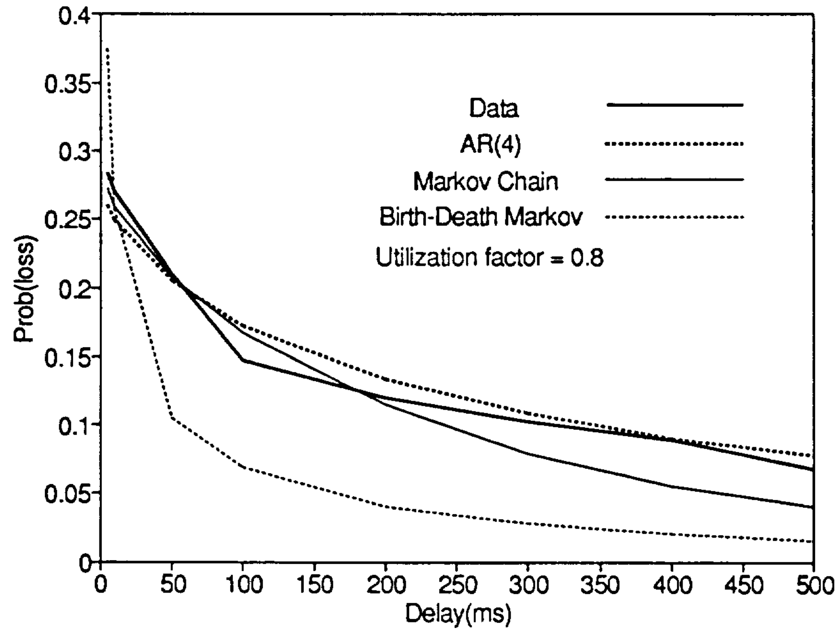


Figure 14 Cell loss probability of several models (Sequence 1, Case 2).

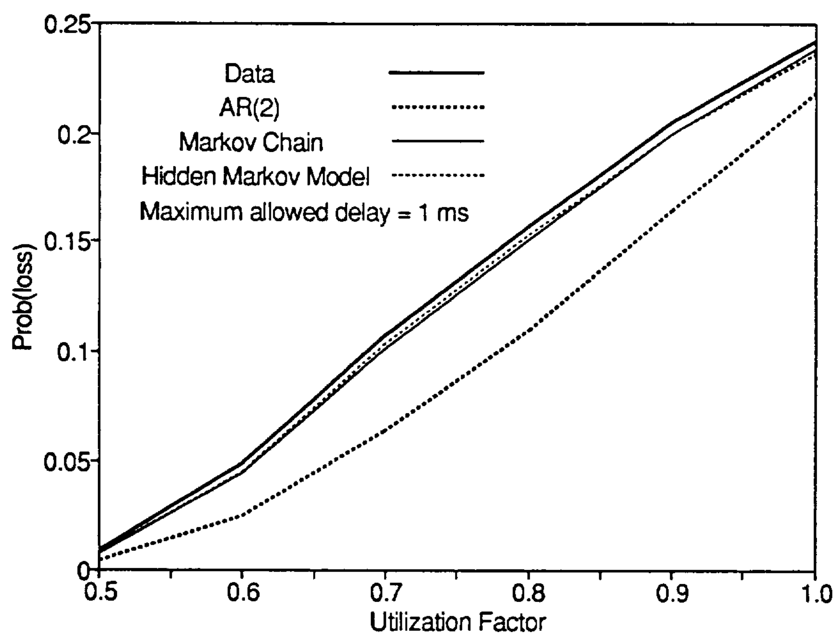


Figure 15 Cell loss probability of several models (Sequence 2, Case 1).

	Cell Loss Probability for Various Utilization Factor				
	1.0	0.9	0.8	0.7	0.6
Data	0.2424	0.2046	0.1577	0.1067	0.0491
AR(2)	0.2176±0.0205	0.1642±0.0187	0.1100±0.0159	0.0635±0.0127	0.0245±0.0085
MC	0.2383±0.0195	0.1995±0.0166	0.1516±0.0133	0.1006±0.0101	0.0442±0.0060
HMM	0.2360±0.0210	0.1992±0.0177	0.1535±0.0135	0.1029±0.0087	0.0451±0.0035

Table 3 Cell loss probability of several models (Sequence 2).

alternatively. Long period of high bit-rate data will dominate the probability of cell loss apparently, if transmission rate is less than the average of high bit-rate period. Markov chain model has the ability to generate collection of large frames because it follows the state transition probability. Hidden Markov model also can produce clumps of large frames since it has "hidden" high bit-rate state. In the simulation, utilization factor is a variable and delay constraint is set to be 1 ms. It is shown clearly, from Figure 15, that Markov chain model and HMM closely follow the performance of recorded data but AR(2) model fails to do that. AR(2) model greatly underestimates the probability of cell loss since it was not able to produce consecutive large frames. Table 3 shows the simulation along with *t*-test results.

5 Conclusion

It has been shown that AR model with moderate order can properly model homogeneous video ssource. Markov chain model has excellent simulation performance for both sequences but without analysis significance. Minisource model is basically a two-phase burst/silence model, which is used extensively in speech source modelling. Aggregate minisource model with birth-death property can model smooth process and is analytically traceable. In the study of statistical multiplexing, aggregate minisource is an accurate model because multiplexed source is smoothed out according to the rule of large number. HMM is believed to be a good simulation model since it can handle complicated sequence with underlying stochastic process. It could be realistic for a long video transmission. Finally, it is noted that the effect of packetization is not ignored in the simulation since we did not adopt fluid flow approximation. Congestion control for video transmission in networks will be emphasis for our future research, simulation will be run with models which are justified in this report.

References

[1] M. Nomura, T. Fujii, and N. Ohta, "Basic characteristics of variable bit rate video coding in ATM environment," *IEEE J. Selected Areas Commun.*, vol. 7, no. 5, pp. 752-760, June 1989.

[2] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson, and J. Robbins, "Performance models of statistical multiplexing in packet video communications," *IEEE Trans. Commun.*, vol. COM-36, no. 7, pp. 834-843, July 1988.

[3] D. P. Heyman, A. Tabatabai, and T. V. Lakshman, "Statistical analysis and simulation study of video teleconference traffic in ATM Networks," *IEEE*

Trans. Circuits and Systems for Video Tech., vol. 2, no. 1, pp. 49-58, March 1992.

[4] P. Sen, B. Maglaris, N. Rikli, and D. Anastassiou, "Models for packet switching of variable bit-rate video sources," *IEEE J. Selected Areas Commun.*, vol. 7, no. 5, pp. 865-869, June 1989.

[5] G. Ramamurthy, and B. Sengupta, "Modeling and analysis of a variable bit rate video multiplexer," *7th Specialist Seminar, International Teletraffic Congress*, Morristown, NJ, Oct. 1990.

[6] B. Melamed, D. Raychaudhuri, B. Senguta, and J. Zdepski, "TES-based traffic modeling for performance evaluation of integrated networks," *INFOCOM'92*, pp. 75-84.

[7] P. J. Brockwell, and R. A. Davis, *Time Series: Theory and Methods*. New York: Springer-Verlag.

[8] L. Kleinrock, *Queueing Systems Volume I: Theory*. New York: Wiley-Interscience.

[9] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257-286, Feb. 1989.

[10] A. M. Mood, F. A. Graybill, and D. C. Boes, *Introduction to the Theory of Statistics*, New York: McGraw-Hill.

[11] H. Yamada, and S. Sumita, "A traffic measurement method and its application for cell loss probability estimation in ATM networks," *IEEE J. Selected Areas Commun.*, vol. 9, no. 3, pp. 315-324, Apr. 1991.

A JOINT SOURCE/CHANNEL CODER DESIGN*

Khalid Sayood, Fuling Liu and Jerry D. Gibson

Khalid Sayood, Dept. of Electrical Engineering, University of Nebraska, Lincoln, NE
Fuling Liu, Western Atlas Company, Houston, Texas
Jerry D. Gibson, Dept. of Electrical Engineering, Texas A&M University, College Station, TX

ABSTRACT. We examine the situation where there is residual redundancy at the source coder output. We have previously shown that this residual redundancy can be used to provide error correction without a channel encoder. In this paper we extend this approach to conventional source coder/convolutional coder combinations. We also develop a design for nonbinary encoders for this situation. We show through simulation results that the proposed systems consistently outperform conventional source-channel coder pairs with gains of greater than 10dB at high probability of error.

1 Introduction

One of Shannon's many fundamental contributions was his result that source coding and channel coding can be treated separately without any loss of performance for the overall system [1]. The basic design procedure is to select a source encoder which changes the source sequence into a series of independent, equally likely binary digits followed by a channel encoder which accepts binary digits and puts them into a form suitable for reliable transmission over the channel. However, the separation argument no longer holds if either of the following two situations occur:

i. The input to the source decoder is different from the output of the source encoder, which happens when the link between the source encoder and source decoder is no longer error free, or

ii. The source coder output contains redundancy.

Case (i) occurs when the channel coder does not achieve zero error probability and case (ii) occurs when the source encoder is suboptimal. These two situations are common occurrences in practical systems where source or channel models are imperfectly known, complexity is a serious issue, or significant delay is not tolerable. Approaches developed for such situations are usually grouped under the general heading of joint source/channel coding.

Most joint source channel coding approaches can be classified in two main categories; (A) approaches which entail the modification of the source coder/decoder structure to reduce the effect of channel errors, [2-18] and (B) approaches which examine the distribution of bits between the source and channel coders [19-21]. The first set of approaches can be divided still further into two classes. One class of approaches examines the modification of the overall structure [2-10], while the other deals with the modification of the decoding procedure to take advantage of the redundancy in the source coder output.

In this paper we present an approach to joint source/channel coder design, which belongs to category A, and hence we explore a technique for designing joint source/channel coders, rather than ways of distributing bits between source coders and channel coders. We assume that the two nonideal situations referred to earlier are present. For a nonideal source coder, we use MAP arguments to design a decoder which takes advantage of redundancy in the source coder output to perform error correction. We have previously shown that this approach can provide error protection at high error rates [16, 17]. In this paper we show that the use of such a decoder in conjunction with a channel encoder can provide excellent error protection over a wide range of channel error probabilities. We then use the decoder structure to infer a channel encoder structure which is similar to a nonbinary convolutional encoder.

2 The Design Criterion

For a discrete memoryless channel (DMC), let the channel input alphabet be denoted by $A = \{a_0, a_1, \ldots, a_{M-1}, \}$, and the channel input and output sequences by $Y = \{y_0, y_1, \ldots, y_{L-1}\}$ and $\hat{Y} = \{\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{L-1}\}$, respectively. If $\mathcal{A} = \{A_i\}$ is the set of sequences $A_i = \{\alpha_{i,0}, \alpha_{i,1}, \ldots, \alpha_{i,L-1}\}$, $\alpha_{i,k} \epsilon A$, then the optimum receiver (in the sense of maximizing the probability of making a correct decision) maximizes $P[C]$, where

$$P[C] = \sum_{A_i} P[C|\hat{Y}]P[\hat{Y}].$$

This in turn implies that the optimum receiver maximizes $P[C|\hat{Y}]$. When the receiver selects the output to be A_k, then $P[C|\hat{Y}] = P[Y = A_k|\hat{Y}]$. Thus, the optimum receiver selects the sequence A_k such that

$$P[Y = A_k|\hat{Y}] \geq P[Y = A_i|\hat{Y}] \ \forall_i.$$

Noting that

$$P(Y|\hat{Y}) = \frac{P(\hat{Y}|Y)P(Y)}{P(\hat{Y})}$$

and for fixed length codes $P(\hat{Y})$ is irrelevant to the receiver's operation [22], the optimal receiver maximizes $P(\hat{Y}|Y)P(Y)$. If we impose a first order Markov assumption on $\{y_i\}$, we can easily show that

$$P\left(\hat{Y}|Y\right) P\left(Y\right) = \prod P\left(\hat{y}_i|y_i\right) P\left(y_i|y_{i-1}\right)$$

This result addresses the situation in case (ii), i.e., the situation in which the source coder output (which is also the channel input sequence) contains redundancy. Using this result, we can design a decoder which will take advantage of dependence in the channel input sequence. The physical structure of the decoder can be easily obtained by examining the quantity to be maximized. The optimum decoder maximizes $P(\hat{Y}|Y)P(Y)$ or equivalently $\log P(\hat{Y}|Y)P(Y)$, but

$$\log P(\hat{Y}|Y)P(Y) = \sum \log P(\hat{y}_i|y_i)P(y_i|y_{i-1}) \qquad (1)$$

which is similar in form to the path metric of a convolutional decoder. Error correction using convolutional codes is made possible by explicitly limiting the possible codeword to codeword transitions, based on the previous code input and the coder structure. At the receiver the decoder compares the received data stream to the *a priori* information about the code structure. The output of the decoder is the sequence that is most likely to be the transmitted sequence. In the case where there is residual strucure in the source coder output, the structure makes some sequences more likely to be the transmitted

*This work was supported in part by NASA Lewis Research Center (NAG 3-806) and NASA Goddard Space Flight Center (NAG 5-916)

0-7803-0950-2/93/$3.00©1993IEEE

sequence, given a particular received sequence. In other words, even when there is no structure being imposed by the encoder, there is sufficient residual structure in the source coder output that can be used for error correction. The structure is reflected in the conditional probabilities, and can be utilized via the path metric in (1) in a decoder similar in structure to a convolutional decoder. However, to implement this decoder we need to be able to compute the path metric.

Examining the branch metric, we see that it consists of two terms $\log P(\hat{y}_i|y_i)$ and $\log P(y_i|y_{i-1})$. The first term depends strictly on our knowledge of the channel. The second term depends only on the statistics of the source sequence. In our simulation results we have assumed that the channel is a binary symmetric channel with known probability of error. We have obtained the second term using a training sequence.

In [17] we showed that the use of the decoder led to dramatic improvements under high error rate conditions. However at low error rates the performance improvement was from nonexistent to minimal. This is in contrast to standard error correcting approaches, in which the greatest performance improvements are at low error rates, with a rapid deterioration in performance at high error rates. In this work we combine the two approaches to develop a joint source channel codec which provides protection equal to the standard channel encoders at low error rates while also providing significant error protection at high error rates.

3 Convolutional Encoders and Joint Source/Channel Decoder

As convolutional coders provide excellent error protection at low error rates, and have a decoder structure similar to the JSC decoder, one way we can combine the two approaches is to obtain the transition probabilities of the convolutional encoder output and use the Joint Source/Channel (JSC) decoder described above instead of the conventional convolutional decoder. The convolutional decoder uses the structure imposed by the encoder and the Hamming metric to provide error protection. The decoder does not use any of the residual structure from the source coder output. We can make use of the residual structure by noting that the path labels transmitted by the convolutional encoder comprise the channel input alphabet $\{y_i\}$. We can then use a training sequence to obtain the transition probabilities $P(y_i|y_{i-1})$, and an estimate of the channel error probability to obtain $P(\hat{y}_i|y_i)$. These can be used to compute the branch metric L which can be used instead of the Hamming metric in the decoder.

We simulated this approach using a two bit DPCM system as the source encoder. We used the two images shown in Figure 1 as the source. The USC Girl image was used for training (obtaining the requisite transition probabilities) and the USC Couple image for testing. The output of the DPCM system was encoded using a (2,1,3) convolutional encoder with connection vectors

$$g^{(1)} = 64 \quad g^{(2)} = 74.$$

The convolutional encoder was obtained from [24]. The performance of the different systems was evaluated using two different measures. One was the reconstruction signal-to-noise ratio (RSNR) defined as

$$RSNR = 10\log_{10}\frac{\sum u_i^2}{\sum (u_i - \hat{u}_i)^2}$$

where u_i is the input to the source coder (source image) and $\hat{u}_i$ is the output of the source decoder (reconstructed image). The other performance measure was the decoded error probability. The received sequence was decoded using either a standard convolutional decoder or the JSC decoder. A block diagram of the system is shown in Figure 2. The results are presented in Figure 3. While there is some improvement in the decoded error probability for high error rates, the RSNR actually goes down

Figure 1. Images used in the simulation

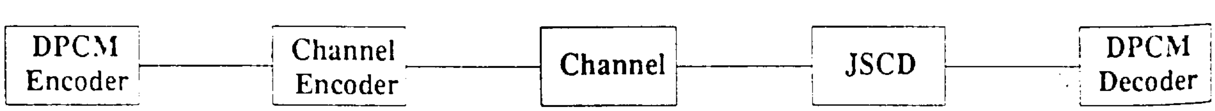


Figure 2. **Block Diagram of Proposed System**

ORIGINAL PAGE IS
OF POOR QUALITY

...the MAP decoded sequence. This is somewhat disappointing until one realizes that the JSC decoder makes use of the structure in the nonbinary output of the source coder. When

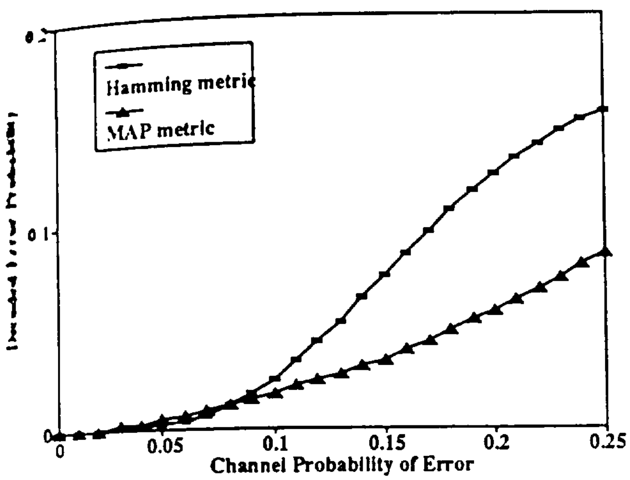


Figure 3a. Decoded error probability for the (2,1,3) convolutional coder.

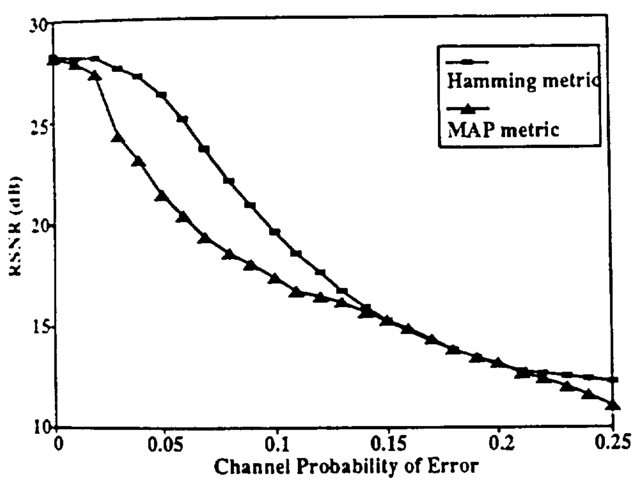


Figure 3b. RSNR for the (2,1,3) convolutional coder

we used the (2,1,3) coder we destroyed some of this structure because the source coder puts out two bit words while the channel coder codes the input one bit at a time. Therefore, if we could preserve the structure in the source coder output by coding the two bit words as a unit we should get improved performance. To verify this we conducted another set of simulation with a rate 1/2 (4,2,1) convolutional code with connection vectors

$$g_1^{(1)} = 6 \quad g_1^{(2)} = 0 \quad g_1^{(3)} = 6 \quad g_1^{4} = 4$$
$$g_2^{(1)} = 0 \quad g_2^{(2)} = 6 \quad g_2^{(3)} = 4 \quad g_2^{(4)} = 2.$$

In this case there is a one-to-one match between the source coder output and the channel coder input, and the results shown in Figure 4 reflect this fact. There is considerable improvement in the decoded error probability and there is about a 5 dB improvement obtained by using the MAP decoder at a probability of error of 0.1. These results justify the contention that for best use of the JSC decoder the input alphabet size of the channel coder should be the same as the size of the output alphabet of the source coder. To this point we have been using a MAP decoder with an encoder designed to maximize performance with a Hamming metric. In the next section we propose a general channel coder design to go with the map decoder which has the added flexibility of being able to match the size of the source coder output alphabet.

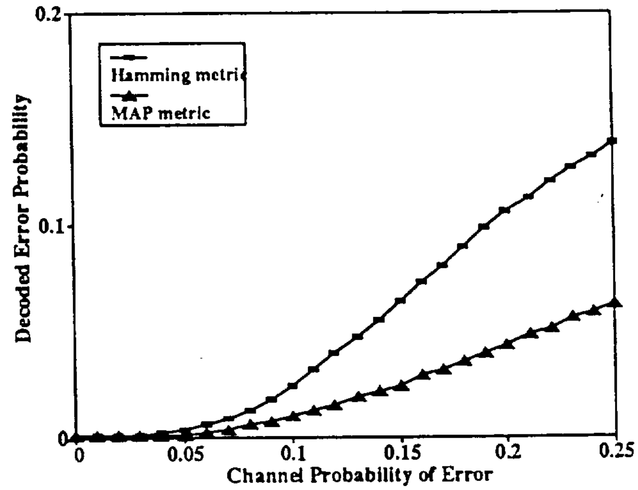


Figure 4a. Decoded error probability for the (4,2,1) convolutional coder

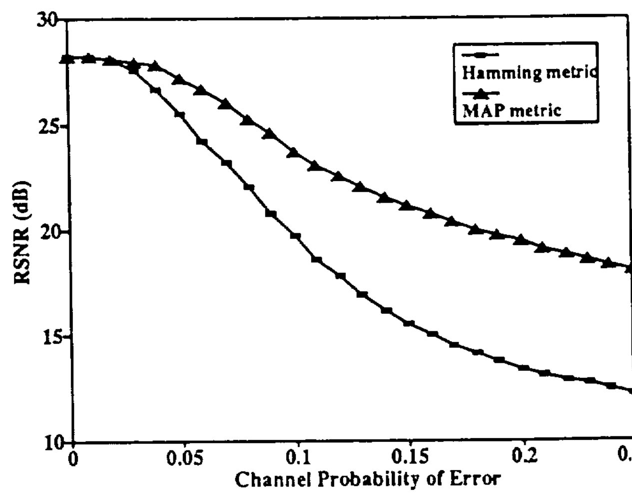


Figure 4b. RSNR for the (4,2,1) convolutional coder

4 A Modified Convolutional Encoder

Given that the preservation of the structure in the source coder output requires the channel coder input alphabet to have a one-to-one match with the generally nonbinary source coder, we propose a general nonbinary convolutional encoder (NCE) whose input alphabet has the requisite property.

Let x_n, the input to the NCE, be selected from the alphabet $A = \{0, 1, 2, ..., N-1\}$, and let y_n, the output alphabet of the NCE, be selected from the alphabet $S = \{0, 1, 2, ..., M-1\}$. Then the proposed NCEs can be described by the following mappings

Rate 1/2 NCE: $M = N^2$; $y_n = Nx_{n-1} + x_n$
Rate 1/3 NCE: $M = N^3$; $y_n = N^2x_{n-2} + Nx_{n-1} + x_n$
Rate 2/3 NCE: $M = N^3$; $y_n = N^2x_{2n-2} + Nx_{2n-1} + x_{2n}$

Because of lack of space we will only describe and present the results for the rate 1/2 NCE. The description and results for the other cases can be found in [25] and are similar to the results for the rate 1/2 NCE code.

The number of bits required to represent the output alphabet of the NCE codes using a fixed length code is

$$\lceil \log_2(M) \rceil = \lceil \log_2(N^2) \rceil = \lceil 2\log_2(N) \rceil$$

Therefore in terms of rate, the rate 1/2 NCE coder is equivalent to a rate 1/2 convolutional encoder. The encoder memory in bits is $2\lceil \log_2(N) \rceil$ as each output value depends on two input values.

As an example, consider the situation when $N = 4$. Then $A = \{0,1,2,3\}$ and $S = \{0,1,2,...,15\}$. Given the input sequence x_n : 0 1 3 0 2 1 1 0 3 3 and assuming the encoder is initialized with zeros, the output sequence will be y_n : 0 1 7 12 2 9 5 4 3 15.

The encoder memory is four bits. Notice that while the encoder output alphabet is of size N^2, at any given instant the encoder can only emit one of N different symbols as should be the case for a rate 1/2 convolutional encoder. For example if $y_{n-1} = 0$, then y_n will take on a value from $\{0,1,2,...,(N-1)\}$. In general, given a value for y_{n-1}, y_n will take on a value from $\{\alpha N, \alpha N+1, \alpha N+2, ..., \alpha N+N-1\}$, where $\alpha = y_{n-1}(mod N)$. This structure can be used by the decoder to provide error protection. The encoder is shown in Figure 5.

4.1 Binary Encoding of the NCE Output

We will make use of the residual structure in the source coder output (which is preserved in the NCE output) at the receiver. However, we can also make use of this structure in selecting binary codes for the NCE output. An intelligent assignment of binary codes can improve the error correcting performance of the system. When each allowable sequence is equally likely, there is little reason to prefer one particular assignment over others. However, when certain sequences are more likely to occur than others, it would be useful to make assignments which increase the 'distance' between likely sequences. While, for small alphabets it is a simple matter to assign the optimum binary codewords by inspection, this becomes computationally impossible for larger alphabets. We use a rather simple heuristic which, while not optimal, provides good results.

Our strategy is to try to maximize the Hamming distance between codewords that are likely to be mistaken for one another. First we obtain a partition of the alphabet based on the fact that given a particular value for y_{n-1}, y_n can only take on values from a subset of the full alphabet. To see this, consider the rate 1/2 NCE; then the alphabet S can be partitioned into the following sub-alphabets:

$$S_0 = (0, 1, 2, 3..., N-1)$$
$$S_1 = (N, N+1, ..., 2N-1)$$
$$\vdots$$
$$S_{N-1} = (N(N-1), N(N-1)+1, ..., N^2-1)$$

where the encoder will select letters from alphabet S_j at time n if $j = y_{n-1}(\text{mod} N)$. Now for each sub-alphabet we have to pick N codewords out of M $(= N^2)$ possible choices. We first pick the sub-alphabet containing the most likely letter. The letters in the sub-alphabet are ordered according to their probability of occurrence. We assign a codeword a from the list of available codewords to the most probable symbol. Then, assign the complement of a to the next symbol on the list. Therefore the distance between the two most likely symbols in the list is $K = \lceil \log_2 M \rceil$ bits. We then pick a codeword b from the list which is at a Hamming distance of $K/2$ from a and assign it and its complement to the next two elements on the list. This process is continued with the selection of letters that are $K/2^k$ away from a at the k^{th} step until all letters in the subalphabet have a codeword assigned to them.

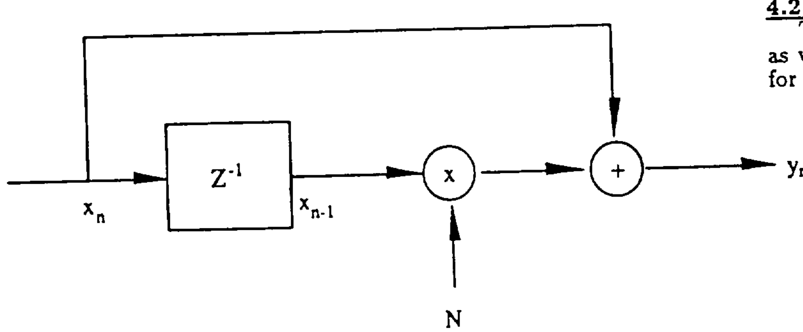


Figure 5. Rate 1/2 nonbinary Convolutional Encoder

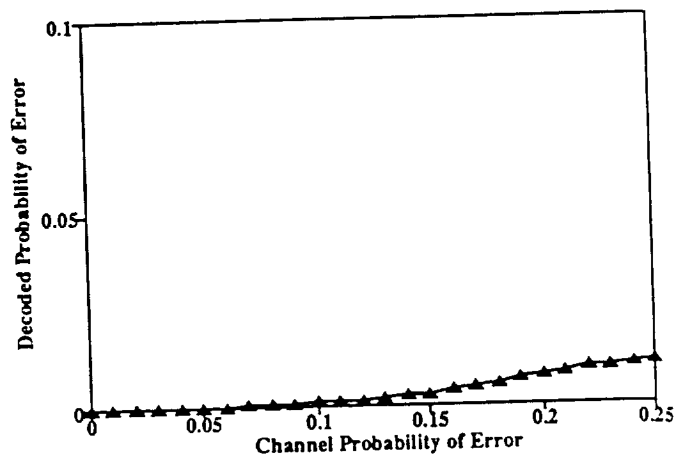


Figure 6a. Decoded vs channel error for rate 1/2 NCE

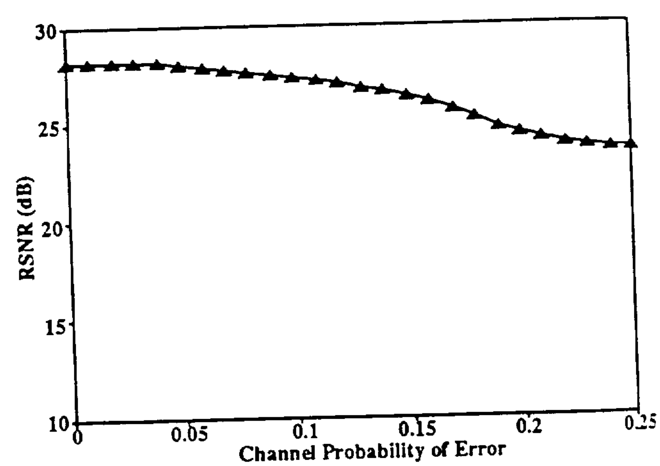


Figure 6b. RSNR for rate 1/2 NCE coder vs channel error

4.2 Simulation Results

The proposed approach was simulated using the same setup as was used in the preceding simulations. We show the results for the rate 1/2 NCE coder in Figure 6 and comparisons in

ORIGINAL PAGE IS OF POOR QUALITY

e 7. Note the dramatic improvement in performance with ate 1/2 NCE code. At a probability of error of 0.1 the R drops by less than 1 dB. For the same channel conditions se of the (2,1,3) code results in a drop of more than 10 ooking at the decoded error probabilities, even when the nel error probability is 0.25, the decoded error probability s than 10^{-2}. This improvement has been obtained with a minimal increase in complexity. Similar results have also obtained for rate 1/3 and 2/3 NCE codes.

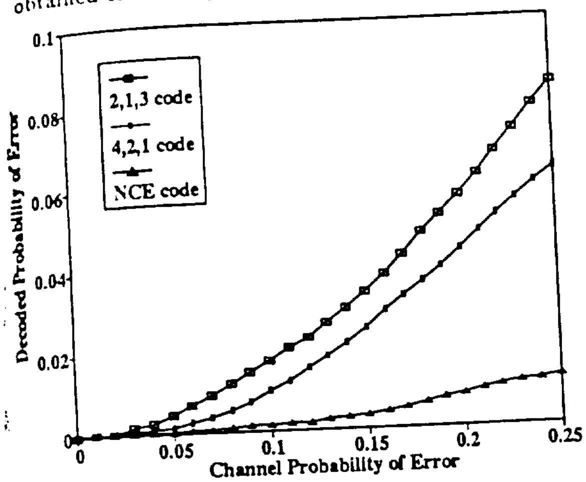


Figure 7a. Decoder error probability for the three MAP decoded systems

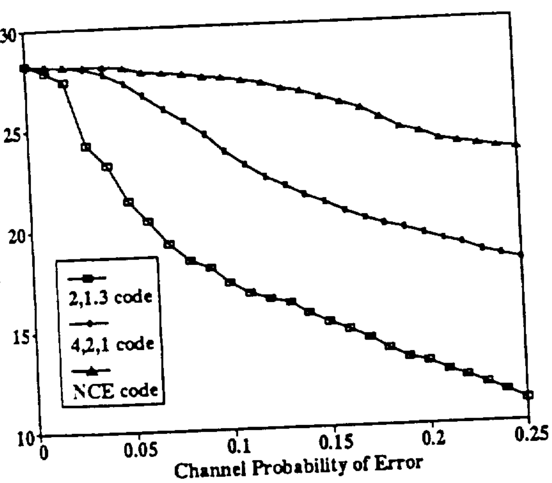


Figure 7b. RSNR vs channel error for the three MAP decoded systems

5 A Modified Convolutional Encoder

If the source and channel coder are designed in a "joint" manner, that is the design of each takes into account the overall conditions (source as well as channel statistics), we can obtain excellent performance over a wide range of channel conditions. In this paper we have presented one such design. The resulting performance improvement seems to validate this approach, with a "flattening out" of the performance curves. This flattening out of the performance curves makes the approach useful for a large variety of channel error conditions.

References

[1] C. E. Shannon, *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, 1948.

[2] J. G. Dunham and R. M. Gray, *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 516–519, July 1981.

[3] E. Ayanoğlu and R. M. Gray, *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 855–865, Nov. 1987.

[4] J. G. Dunham and R. M. Gray, *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 516–519, July 1981.

[5] J. L. Massey, in *Communication Systems and Random Process Theory*, J. K. Skwirzynski, ed., Sijthoff and Nordhoff: Netherlands, pp. 279–293, 1978.

[6] T. C. Ancheta, Jr. Ph.D. dissertation, Dept. of Electrical Engr., Univ. of Notre Dame. Aug. 1977.

[7] K-Y Chang and R. W. Donaldson, *IEEE Trans. Commun.*, vol. COM-20, pp. 338–350, June 1972.

[8] A. J. Kurtenbach and P. A. Wintz, *IEEE Trans. Commun. Technol.*, vol. COM-17, pp. 291–302, April 1969.

[9] N. Farvardin and V. Vaishampayan, *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 827–838, November 1987.

[10] D. J. Goodman and C. E. Sundberg, *Bell Syst. Tech. J.*, vol. 62, pp. 2017–2036, Sept. 1983.

[11] D. J. Goodman and C. E. Sundberg, *Bell Syst. Tech. J.*, vol. 62, pp. 2735–2764, Nov. 1983.

[12] R. C. Reininger and J. D. Gibson, *IEEE Trans. Commun.*, vol. COM-31, pp. 572–577, April 1983.

[13] R. Steele, D. J. Goodman, and C. A. McGonegal, *IEEE Trans. Commun.*, vol. COM-27, pp. 252–255, January 1979.

[14] R. Steele, D. J. Goodman, and C. A. McGonegal, *Elec. Lett.*, vol. 13, pp. 351–353, June 1977.

[15] K. N. Ngan and R. Steele, *IEEE Trans. Commun.*, vol. COM-30, pp. 257–269, January 1982.

[16] K. Sayood and J. C. Borkenhagen, *Proceedings IEEE International Conference on Communications*, June 1986, pp. 1888–1892.

[17] K. Sayood and J. C. Borkenhagen, *IEEE Transactions on Communications*, vol. COM-39, June 1991.

[18] K. Sayood and J. D. Gibson, *Proc. 22nd Annual Conference on Information Sciences and Systems*, Princeton, NJ, Mar. 1988, pp. 380-385.

[19] J. W. Modestino, D. G. Daut, and A. L. Vickers, *IEEE Trans. Commun.*, vol. COM-29, pp. 1262–1274, Sept. 1981.

[20] D. Comstock and J. D. Gibson, *IEEE Trans. Commun.*, vol. COM-32, pp. 856–861, July 1984.

[21] C. C. Moore and J. D. Gibson, *IEEE Trans. Commun.*, vol. COM-32, August 1984.

[22] N. Phamdo and N. Farvardin, private correspondence.

[23] K. Sayood, J. D. Gibson, and F. Liu, *Proc. 22nd Annual Asilomar Conference on Circuits, Systems, and Computers*, Nov. 1988, pp. 102-106.

[24] S. Lin and D. J. Costello, *Error Control Coding*, Prentice Hall, 1983.

[25] K. Sayood, F. Liu and J. D. Gibson, *IEEE Trans. Commun.* To be submitted.

COMPRESSION OF COLOR-MAPPED IMAGES*

A.C. Hadenfeldt and K. Sayood

A.C. Hadenfeldt, University of Nebraska Medical Center, Omaha, Nebraska
K. Sayood, Dept. of Electrical Engineering, Univ. of Nebraska, Lincoln, NE

ABSTRACT In a standard image coding scenario, pixel-to-pixel correlation nearly always exists in the data, especially if the image is a natural scene. This correlation is what allows predictive coding schemes (e.g., DPCM) to perform efficient compression. In a color-mapped image, the values stored in the pixel array are no longer directly related to the pixel intensity. Two color indices which are numerically adjacent (close) may point to two very different colors. The correlation still exists, but only via the colormap. This fact can be exploited by sorting the color map to reintroduce the structure. In this paper we study the sorting of colormaps and show how the resulting structure can be used in both lossless and lossy compression of images.

1 Introduction

Many lower-cost image display systems use color-mapped (or pseudo-color) displays. While there has been considerable attention devoted to the compression of monochrome and full-color images, the compression of color-mapped images has not received similar attention.

The human eye can distinguish hundreds of thousands of different colors in a color space, depending on viewing conditions [1]. A full-color (also called true-color) frame buffer provides a means of displaying this wide range. Such a system is illustrated in Figure 1.

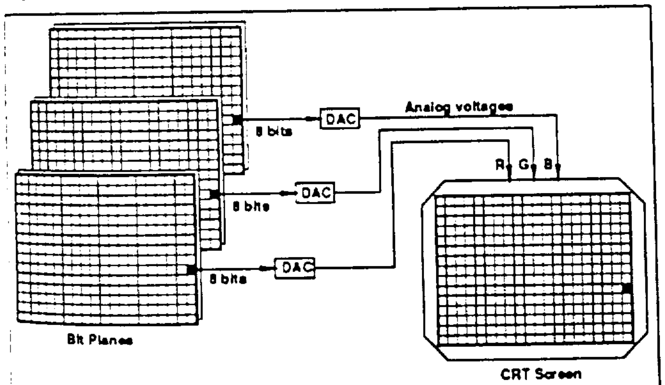


Figure 1 Full-color frame buffer

Many applications of digital images benefit from, or require, color capabilities to be effective. If a full-color display is used, an

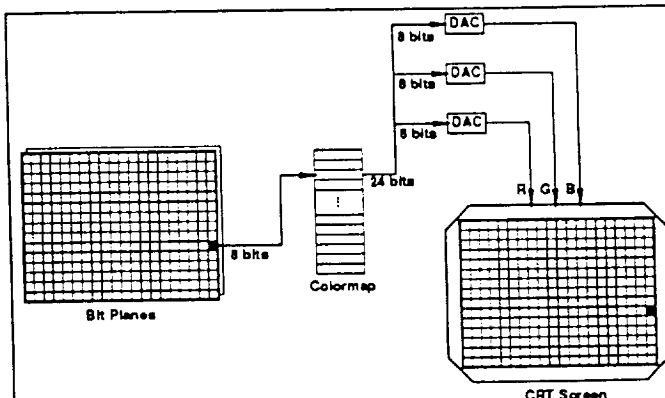


Figure 2 Color-mapped frame buffer

application may become too costly to implement practically. Also, the images involved require large amounts of storage space, whether in display memory or on a mass-storage device. A less expensive solution is needed.

These applications naturally lead to the pseudo-color or color-mapped frame buffer, shown in Figure 2. This type of display is typical of those found on personal computers and workstations. A smaller amount of image memory is required, one-third that of the full-color system example. The values stored in memory are used as indices into a 24-bit table, the colormap.

Each entry in the colormap consists of 8-bit values for the red, green, and blue portions of the pixel. These three values are then passed through DACs to the red, green, and blue electron guns of the CRT, as with full-color system. The color-mapped system allows the display of a small number of colors at a time, 2^8 for the system shown in the figure, which can be selected from a larger set of colors (2^{24} for this example). By careful selection of the colors in the colormap, a large variety of images can be displayed, often with quality approaching that of a full-color display system.

The use of the colormap, however, disguises the spatial structure in the image. An indication of this can be obtained by calculating the zero-th and first order entropies of the image. These quantities were computed using the index arrays for the four test images shown in Figure 3, and are listed in Table 1.

Figure 3a Park　Figure 3b. Lena　Figure 3c Lincoln　Figure 3d Omaha

Figure 3 Test images

* This work was supported by the NASA Goddard Space Flight Center (NAG 5-1612) and the NASA Lewis Research Center (NAG 3-806).

Table 1 Entropies of the Source Images

Image	H_0	H_1
Lena	7.617	7.413
Park	7.470	7.797
Omaha	7.242	7.165
Lincoln	5.916	6.674

The large values of H_1 in the table verify that the spatial correlation in the image pixel values has been reduced by the color-mapping process. The values of H_0 are also relatively large, a direct result of selecting an 8-bit colormap. The data compression due to the color quantization process implies that the color index values stored in the image are more critical than similar values in, for example, an achromatic image. To further verify this, an experiment was conducted in which errors were introduced in the least significant bit of a color-mapped image, similar to what might be encountered if the color index values were quantized. The resultant images were of poor subjective quality at best, and often completely unrecognizable. Since quantization is a part of many popular source coding schemes, the available choices for compression schemes become limited.

2 Colormap Sorting

In the previous section, several problems unique to color-mapped images were discussed. The root of these problems is that the colormap indices stored in the image have little relationship with each other, which complicates coding for progressive transmission. In this section, methods of restoring this relationship are discussed.

Colormap sorting is a combinatorial optimization problem. Treating the K colormap entries as vectors, the problem is defined as follows. Given a set of vectors $\{a_1, a_2, ..., a_K\}$ in a three-dimensional vector space and a distance measure $d(i,j)$ defined between any two vectors a_i and a_j, find an ordering function $L(k)$ which minimizes the total distance D:

$$D = \sum_{k=1}^{K-1} d(L(k),\ L(k+1)) \qquad (1)$$

The ordering function L is constrained to be a permutation of the sequence of integers $\{1, ..., K\}$. Another possibility results when the list of colormap entries is considered as a ring structure. That is, the colormap entry specified by $L(K)$ is now considered to be adjacent to the entry specified by $L(1)$. In this case, an additional term of $d(L(K), L(1))$ is added to the distance formula D.

The sorting problem is similar to the well-known travelling salesman problem, and is identical if the colormap is considered as a ring structure. As such, the problem is known to be NP-complete [3], and the number of possible orderings to consider is $1/2[(K-1)!]$[4]. Algorithms exist which can solve the problem exactly [4][5]; however, these algorithms are computationally feasible only for K no greater than about 20. Efficient algorithms for locating a local minimum exist [4] for $K \leq 145$. For large colormaps such as $K = 256$, another approach is necessary. Simulated annealing was chosen as the sorting method for the colormaps in this work, and is described in more detail in Section 2.1.

The distance metric d was chosen to be (unweighted) Euclidean distance, and different color spaces were investigated. Three color spaces were selected: the NTSC RGB space, the CIE L*a*b* space, and the CIE L*u*v* space. The NTSC RGB space was chosen since it corresponds to the color primaries of the original images. Color spaces which can be linearly transformed to the NTSC RGB space were not considered, since the use of an unweighted Euclidean distance measure would give similar results for such a color space. The two CIE color spaces were selected since they provide a means to measure perceptual color differences.

2.1 Sorting Using Simulated Annealing

Simulated annealing [3][6] is a stochastic technique for combinatorial minimization. The basis for the technique comes from thermodynamics and observations concerning the properties of materials as they are cooled. The technique described in this section is based on the implementation in [6].

To illustrate this concept, consider an iron block. At high temperatures, the iron molecules move freely with respect to each other. If the block is *quenched* (cooled very quickly), the molecules will be locked together in a high-energy state. On the other hand, if the block is *annealed* (cooled very slowly), the molecules will tend to redistribute themselves as they lose energy, with the result being a lower energy lattice which is much stronger. The distribution of molecular energies is characterized by the Boltzmann distribution:

$$P(E) \sim e^{-E/kT} \qquad (2)$$

where E is the energy state, T is the temperature, and k is Boltzmann's constant. In a combinatorial optimization situation, the Boltzmann distribution can be used to temporarily allow increases in the cost function, while still generally striving to achieve a minimum.

Solving the colormap sorting problem involves selecting each color only once while minimizing the sum of the distances between the colors. To find a solution using simulated annealing, an initial path through the nodes (colors) is chosen, and its cost computed. The algorithm then proceeds as follows:

1. Select an initial temperature T and a cooling factor α.
2. Choose a temporary new path by perturbing the current path (see below), and compute the change in path cost, $\Delta E = E_{new} - E_{old}$. If $\Delta E \leq 0$, accept the new path.
3. If $\Delta E > 0$, randomly decide whether or not to accept the path. Generate a random number r from a uniform distribution in the range [0, 1), and accept the new path if $r < \exp(-\Delta E/T)$.
4. Continue to perturb the path at the current temperature for I iterations. Then, "cool" the system by the cooling factor: $T_{new} = \alpha T_{old}$. Continue iterating using the new temperature.
5. Terminate the algorithm when no path changes are accepted at a particular temperature.

The decision-making process is known as the *Metropolis algorithm*. Note that the decision process will allow some changes to the path which increase its cost. This makes it possible for the simulated annealing method to avoid easily being trapped in a local minimum of the cost function. Hence, the algorithm is less sensitive to the initial path choice.

For the images of this work, initial values of T ranged from 80 to 500, depending on the color space used. The cooling factor α was usually chosen as 0.9. The simulated annealing algorithm seemed to be most sensitive to the choice of this value, as values outside the range [0.85, 0.95] caused the cooling to occur too slowly or too quickly. The number of iterations per temperature I was chosen as 100 times the number of nodes (colors), or 25,600. However, to improve the execution speed of the algorithm an improvement suggested by [6] was added, which causes the algorithm to proceed to the next temperature if (10)(number of nodes) = 2560 successful path changes are made at a given temperature.

Also, a method for perturbing the path must be selected. In this work, the perturbations were made using the suggestions of Lin [4][6]. At each iteration, one of two possible changes to the path are made, chosen at random. The first is a *path transport*, which removes a segment of the current path and reinserts it at another point in the path. The location of the segment, its length, and the new insertion point are chosen at random. The second perturbation method, called *path reversal*, removes a segment of the current path and reinserts it at the same point in the path, but with the nodes in reverse order. The location and length of the segment are again randomly chosen.

The algorithm outlined in the previous paragraphs formulates colormap sorting as a travelling salesman problem. This type of problem usually assumes a complete tour will be made (i.e., the salesman desires to return to the original city). Hence, the colormap is assumed to have a ring-like structure. However, the simulated annealing technique can also be used if this is not the case, allowing the colormap to be considered as a linear list structure. Experiments using both structures were conducted.

3 Colormap Sorting and Lossless Compression

The results of sorting the colormaps of the test images using simulated annealing are shown in the following tables. Table 2 shows results for sorting the colormap as a circular ring structure, while Table 3 shows the results of sorting the colormap as a linear structure. Given in the tables are values for the resulting first-order entropy and the final path cost (the distance measure D).

Table 2 Resultant Images With Circularly Sorted Colormaps

Image Name	RGB Space		L*a*b Space		L*u*v* Space	
	Cost	H_1	Cost	H_1	Cost	H_1
Lena	13.88	5.641	857.80	5.627	208.49	5.480
Park	19.32	6.325	1609.46	6.330	310.41	6.218
Omaha	11.04	6.209	1081.82	6.303	363.21	6.178
Lincoln	10.62	5.513	1193.88	5.831	224.06	5.478

Table 3 Resultant Images With Linearly Sorted Colormaps

Image Name	RGB Space		L*a*b Space		L*u*v* Space	
	Cost	H_1	Cost	H_1	Cost	H_1
Lena	11.68	5.575	847.29	5.933	200.31	5.512
Park	15.66	6.260	1509.25	6.775	292.29	6.546
Omaha	10.81	6.532	1004.69	6.554	283.66	6.199
Lincoln	10.61	5.774	1177.80	6.120	204.64	5.735

Note that the zero-order entropy H_0 is not changed by the sorting process, since permuting the colormap entries does not change the frequency of occurrence of a particular color. The lower first-order entropies of the resultant images indicate that some of the spatial correlation between color indices has been restored in each case. The sorting results for the NTSC RGB space show that sorting in this space yields good results, if entropy reduction (the first goal stated above) is the goal. However, the L*u*v* space sorting gives better results, with the added advantage that the perceptual differences between colormap entries has been considered. Hence, the resultant images from this sort should also be able to accept quantization errors while maintaining good subjective quality, the second goal stated previously. We examine this further in the next section. In terms of lossless compression, the sorting has resulted in a drop of 2 bits per pixel for the Lena image and 1 to 1.5 bits per pixel for the other images. For a 512x512 image this translates to a savings of between 32,768 to 65,536 bytes per image. For a large database of images this could be a considerable saving.

4 Colormap Sorting and Lossy Compression

The sorting of the colormap restores some perceptual structure to the colormap indices in the sense that indices close in numerical value are also close in some perceptual sense. Therefore it should be possible to introduce errors into the indices without destroying the image. To verify this hypothesis, we dropped the three least significant bits of the L*u*v*-sorted Park images. Good subjective results were obtained using quantization levels down to as low as 5 bits/pixel from the 8-bit original. The sorted colormap shown was sorted as a linear list in L*u*v* space. Figure 4 shows the result of quantizing the Park image to 5 bits/pixel, before and after the colormap has been sorted. A caveat is in order here. While the distance between the eight-bit indices have more perceptual meaning, the sorted colormap image should not be assumed to have the same properties as an eight-bit monochrome image. In some cases, if the distance between the original and reconstructed (compressed and decompressed) indices is large enough, there might be a drastic change in color between those pixels in the original and reconstructed image. In the monochrome case large distances would correspond to changes in shading which might be overlooked by the viewer.

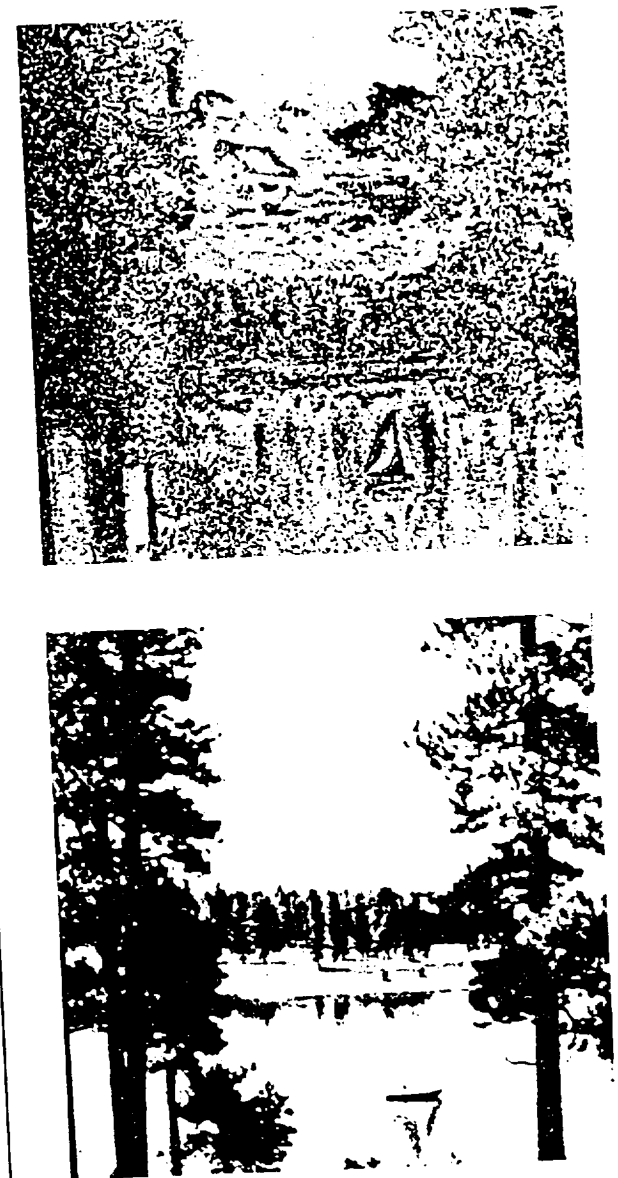


Figure 4 Park image quantized to five bits per pixel with (a) unsorted and (b) sorted colormaps

To see how well the sorted color-mapped images lend themselves to lossy compression we compress them using particular implementations of two popular lossy compression techniques, the Discrete Cosine Transform (DCT) and Differential Pulse Code Modulation (DPCM).

4.1 DCT Coding of Color-mapped Images

In Figure 5 we coded the Lena image with the unsorted colormap at two bits per pixel using the unsorted color map. As can be seen from the figure, the original image is totally lost and all that remains is seemingly random colors. It should be noted that for eight-bit monochrome images, DCT coding at two bits per pixel generally provides a reconstruction which is indistinguishable from the original.

In Figure 6 we show the same image, this time with the sorted color map, coded at two bits per pixel with the fixed bit allocation. [7] The images in Figure 7 were coded at two and one bit per pixel using the JPEG [8] algorithm.[1] Note that while the image coded using the fixed bit allocation shown in Figure 6 is far superior to the image in

[1] The JPEG coded images were coded using software from the independent JPEG group.

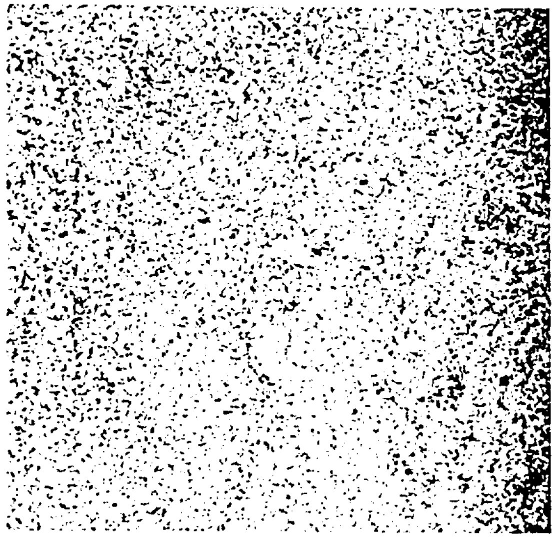


Figure 5 Lena image with unsorted colormap coded at two bits per pixel using JPEG DCT algorithm

Figure 6 Lena image coded at two bits per pixel using DCT with fixed bit allocation

Figure 5, there are still quite a few annoying artifacts. This is because of the nonadaptive nature of the algorithm which, while it minimizes the *average* error, may permit the introduction of large errors in individual blocks. As the color-mapped images are particularly sensitive to large errors, this could account for the low quality reproduction. The JPEG algorithm adapts its bit allocation on a block-by-block basis. Therefore, the image in Figure 7(b) which is coded at half the rate of the image in Figure 6 still provides superior quality.

4.2 DPCM Coding of Color-mapped Images

Standard DPCM coding of color-mapped images is problematic because in the busy regions of images, especially edges, the prediction error is generally large, leading to large overload noise values. In monochrome images these noise values result in a blurred look around edges, which may be acceptable for certain application. However, in color-mapped images these noise values will result in splotches of different colors. The Edge Preserving DPCM (EPDPCM) system

Figure 7 Lena image coded at (a) two bits per pixel and (b) one bit per pixel using JPEG algorithm

avoids this problem by the use of a recursively indexed quantizer [9,10], in which the magnitude of the quantization error is always bounded by $\frac{\Delta}{2}$. This attribute makes it ideal for application to the coding of color-mapped images. Another advantage of the EPDPCM system is that, as the quantizer output alphabet can be kept small without incurring overload error, the output is amenable to entropy coding.

Results using the EPDPCM system are shown in Figure 8. The image in Figure 8(a) was coded at a rate of 2 bits per pixel, while the image in Figure 8(b) was coded with 1.35 bpp.

The advantage of DPCM systems over transform coding systems is their low complexity and higher speed. However, the reconstruction quality obtained using transform coding systems is generally significantly higher than that of DPCM systems at a given rate. Comparing Figure 8(a) and 7(a), this is obviously not the case for the sorted colormapped images. In fact, the quality of the two-bit EPDPCM coded image is actually somewhat higher than the two-bit DCT coded image. Thus using the EPDPCM system provides advantages both in terms of complexity and speed, and reconstruction quality.

Figure 8 Lena image coded at (a) 2 bits per pixel and (b) 1.35 bits per pixel using EPDPCM

5 Conclusion

In this paper we have shown that use of sorted colormaps makes color-mapped images amenable to both lossless and lossy compression. For lossy compression conventional wisdom dictates the use of DCT coding for most types of images. However, for color-mapped images DPCM coding might be more advantageous.

6 References

[1] Foley, J.D., van Dam, A., S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice (Second Edition)*, Reading, MA: Addison-Wesley, 1990.

[2] *Graphics Interchange Format (GIF) Specification*, CompuServe, Inc., Columbus, OH, June 1987.

[3] Aarts, E., and J. Korst, *Simulated Annealing and Boltzmann Machines*, New York: John Wiley and Sons, 1989.

[4] Lin, S., "Computer Solutions of the Traveling Salesman Problem," *Bell System Technical Journal*, pp. 2245–2269, December 1965.

[5] Bellman, R.E., and S.E. Dreyfus, *Applied Dynamic Programming*, Princeton, NJ: Princeton University Press, 1962.

[6] Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C*, New York: Cambridge University Press, 1988.

[7] Jayant, N.S. and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, 1984.

[8] Wallace, G.K., "The JPEG still picture compression standard," *Communications of the ACM*, 34(4):31–44, April 1991.

[9] Rost, M.C. and K. Sayood, "An Edge Preserving Differential Image Coding Scheme," *IEEE Transactions on Image Processing*, 1:250–256, April 1992.

[10] Sayood, K. and S. Na. "Recursively Indexed Quantization of Memoryless Sources," *IEEE Transactions on Information Theory*, IT-38, November 1992.

THE PROPOSED HDTV SCHEMES AND THE NASA NETWORK

Y.C. Chen and K. Sayood
Department of Electrical Engineering
University of Nebraska
Lincoln, NE 68588-0511

Abstract

There are currently several digital HDTV proposals being evaluated by the FCC. There are several advantages to the acceptance of a digital HDTV standard. The technical advantages, which will be of most concern to NASA is the fact that the use of digital technology increases the inter-operability of these systems. The output of a digital system can be relatively easily interfaced with existing transmission facilities, such as the NASA network as described by various CCSDS documents. We are in the processing of simulating the proposed HDTV techniques in the context of using them to transmit HDTV sequences over NASA network. In this paper we present the results of our simulation and discuss the various tradeoffs involved in selecting the different services available on the NASA network.

I. Introduction

High Definition Television (HDTV) has been in the news for most of the decade of the 80's. Most of the early developments in this area took place in Japan and Europe at a time that saw the decline of the consumer electronics industry in the United States. While the first HDTV system was demonstrated by NHK of Japan in 1974, real interest in the United States in HDTV did not occur till the beginning of the 80's. Since then a host of complications have arisen, both technical and political. The technical problems arise from the issues of compatibility both with current US systems and the international market, and the fact that there are now several methods of delivery - terrestrial, DBS, and cable. However, the picture is clearing up with recent FCC decisions on standardization, and there are now several proposals for HDTV which have received FCC approval for testing.

The HDTV analog signal has a nominal bandwidth of around 34 MHz. Given the existing 6 Mhz/channel allocations it would require 6 conventional channels to handle a single HDTV channel. In 1987 the FCC issued a tentative ruling that channel assignments for HDTV signals would be limited to existing VHF and UHF channels. This meant that techniques had to be developed to severely reduce the bandwidth requirements for HDTV. One proposal that was being considered was the use of an "augmentation channel", where one channel would carry NTSC compatible signals, while a second augmentational channel would carry additional information for users with HDTV receivers. However in March 1991, the FCC announced that it would favor approaches which would limit HDTV transmission to 6 MHz channels separate from the channels used for conventional TV transmissions. The implication is that only a single channel be used for HDTV transmission. This announcement more or less forces the proposers to use video compression techniques which can be best used in a digital form. This is reflected in the fact that of the five HDTV proposal currently in contention four are digital schemes.

This move by the FCC has (based on ones point of view) had two positive effects, one political and one technical. The advantage offered by this announcement gives an advantage to US companies, as most of the development in Japan and Europe has been on analog forms of HDTV. Thus the US companies are not at as much a competitive disadvantage with respect to research and development. This is reflected in the fact that the only analog proposal in contention is one from

NHK. The technical advantage, which will be of most concern to NASA is the fact that the use of digital technology increases the inter-operability of these systems. The output of a digital system can be relatively easily interfaced with existing transmission facilities, such as the NASA network as described by various CCSDS documents. Currently we have completed the simulator for the compression aspects of the *Advanced Digital Television* (ADTV) proposed by David Sarnoff Laboratories, NBC, North American Phillips, and Thomson Consumer Electronics. This proposal has also been supported by Texas Instruments. We are at various stages of development in terms of the other HDTV proposals. This paper is organized as follows. First, we give a brief overview of the ADTV system. In Section III, some of the relevant features of the CCSDS recommendations are reviewed. In Section IV, we look at some coding results obtained using ADTV simulator. We compare these results to those obtained using the CCITT H.261 standard. In Section V, we evaluate these results in the context of the CCSDS specifications and make some suggestions as to how the ADTV system could be implemented in the NASA network. Finally, in Section VI the paper is summarized.

II. Advanced Digital Television

There are three key elements in the ADTV system.

- ADTV uses MPEG++(Moving Pictures Expert Group) draft proposal as its compression scheme.
- ADTV incorporates a Prioritized Data Transport (PDT) which is a cell relay-based data transport layer to supports the prioritized delivery of video data. PDT also offers service flexibility and compatibility to broadband ISDN.
- ADTV applies spectral-shaping techniques to Quadrature Amplitude Modulation (QAM) to minimize interference from and to any co-channel NTSC signals.

We have simulated all aspects of the compression algorithm of the ADTV proposal. The compression algorithm as described in the *Advanced Digital Television, System Description* submitted to FCC/ ACATS and as implemented in the simulator is described below[1].

A. Compression Algorithm

The basic compression approach is the MPEG++ algorithm which upgrades the standard MPEG approach to HDTV performance level. The key components of this algorithm are described below.

A.1 Group of Pictures(GOP)

A GOP comprises up to three types of frames, the I, P and B frames. The I frames are processed using only intra-frame DCT coder with adaptive quantization; the P frames are processed using a hybrid temporal predictive DCT coder with adaptive quantization and forward motion compensation; the B frames are processed using a hybrid temporal predictive DCT coder with adaptive quantization and bidirectional motion compensation. The I and P frames are referred to as the anchor frames because of their roles in the bidirectional motion compensation of the B frames.

The GOP structure offers a good tradeoff between the high efficiency of temporal predictive coding, good error-concealment features of periodic intra-only processing, and fast picture acquisition.

A.2 Input Sequencer

The GOP data structure requires some unique sequencing of the input video frames. Because of the backward motion compensation in B frames processing, the anchor frames must be processed before the B frames associated with the two anchors. The frames are transmitted in the same order as they are processed.

A.3 Raster Line to Block/Macroblock Converter

The basic DCT transform unit is an 8 x 8 pixel block called a block. The basic quantization unit is four adjacent blocks of Y, and one U and one V blocks. Such a quantization unit is called a macroblock. The converter converts the raster line format to the block and the macroblock format.

A.4 I-frame Processing

An I frame is processed by an intra-frame DCT coder without motion compensation. A fixed quantizer is applied to the DC coefficient. The AC values are first weighted by a down-loadable quantization matrix before uniform adaptive quantization. The quantization step for the AC coefficients is controlled by a Rate Controller. The I frame coding is pretty much the same as JPEG scheme.

A.5 P-frame Processing

A P frame is first processed by forward motion compensation, motion is always referenced to the nearest past anchor frame. The search area is proportional to the number of B frames between two consecutive anchor frames. The prediction residue or original macroblock, depending on the motion compensation result, goes to DCT coder and quantizer. For intra-macroblocks, the DCT coefficient quantization is identical to that used for the I frames. For motion-compensated macroblocks, the DC and AC coefficients are quantized with same uniform quantizer.

A.6 B-frame Processing

Unlike the P frames, the B frames are subjected to bidirectional motion compensation. The motion references are the two anchor frames sandwiching the B frames. The search regions are proportional to the temporal distance between the B frame and the two anchor frames. Like P-frame macroblock, the B-frame macroblocks have a number of modes. In addition to all the modes for a P-frame macroblock, the B-frame macroblock further includes a bidirectional interpolative mode, using both forward and backward motion compensation, and a unidirectional mode. In the interpolative mode, an average of the forward and the backward motion- compensated macroblocks is used as the prediction macroblock. The B-frame macroblock is processed as a P-frame macroblock.

A.7 Differential, Run-Length and Variable-Length Coding(VLC)

The quantized DC coefficients of all the I-frame macroblock and P-, B-frame macroblocks in intra mode are coded with a DPCM coder. The quantized AC coefficients are coded with VLC after the zigzag scan ordering. Motion vectors are differentially coded. In addition, VLC is applied to all the coded information: motion vectors, macroblock addresses, block types, etc.

B. Data Prioritization Layer

The Prioritization Layer comprises the Priority Processor and the Rate Controller[2].

B.1 Priority Processor

Based on the information from the Rate Controller, the Priority processor pre-calculates the rate of HP(High Priority)/LP(Low Priority) for every frame. HP/LP fractional allocations may vary with the frame type. Every data element gets a priority assignment from the Priority Processor according to its importance. The header is always most important, followed by the motion vector, DC value, low frequency coefficients and high frequency coefficients.

B.2 Rate Controller

The Rate Controller monitors the status of the rate buffers in the Transport Encoder. It uses the buffer occupancy information to compute the necessary compression requirement and feeds the results in the form of appropriate quantization parameters to the Video Processor in the Compression Encoder. The Rate Controller also provides input to the Priority Processor regarding the initial allocation of HP/LP rate for the next Group of Pictures. The algorithm used in this simulation for rate control is given by

$$QS = 2\left\lceil \frac{B}{200p} \right\rceil + 2$$

where QS is the quantizer step-size, and B is a measure of buffer fullness.

C. Transport Layer

The Transport Layer comprises the Transport Processor and the Rate Buffer[2].

C.1 Transport Processor

Data elements are supplied to the Transport Processor from the Prioritization Processor. The Transport Processor generates appropriate header fields for data group. The header fields are used in the construction of a basic transport unit called cell. A cell has a header and a trailer enclosing a payload area. Each cell has a fixed size of 256 bytes long. The header contains chaining and segmentation information which allows data groups to be segmented across cells. This feature limits the propagation of channel error from one cell to the next. The trailer field contains 16-bit error-checking CRC code.

C.2 Rate Buffer

Since the number of cells generated is not constant and the channel coding module interfaces with the Transport Processor at a fixed clock rate, we need a buffer to smooth out any rate variation. The maximum end-to-end delay is dependent on the size of the buffer.

III. NASA Space Network

To speculate how the HDTV service would be accommodated by the NASA network, we briefly review some of the relevant features of the CCSDS recommendations.

A. CCSDS Principal Network

A "CCSDS Principal Network" (CPN) serves as the project data handling network which provides end-to-end data flow in support of the "Experimental", "Observational" and interactive users of Advanced Orbiting Systems. A CPN consists of an "Onboard Network" in an orbiting segment connected through a CCSDS "Space Link Subnetwork" (SLS) either to a "Ground Network" or to another Onboard Network in another orbiting segment. The SLS is the central component of a CPN; it is unique to the space mission environment and provides customized services and data communications protocols. Within the SLS, CCSDS defines a full protocol to achieve "cross support" between agencies. Cross support is defined as the capability for one space agency to bidirectionally transfer another agency's data between ground and space systems using its own transmission resources. A key feature of this protocol is the concept of a "Virtual Channel" which allows one physical space channel to be shared among multiple traffic streams, each of them may have different service requirements. A single physical space channel may therefore be divided into several separate logical data channels, each known as a Virtual Channel (VC).

Eight separate services are provided within a CPN. Two of these services ("Path" and "Internet") operate end-to-end across the entire CPN. They are complementary services, which satisfy different user data communications requirements: some users will interface with only one of them, but many will operate with both. The remaining six services ("Encapsulation", "Multiplexing", "Bitstream", "Insert", "Virtual Channel Data Unit" and "Physical Channel") are provided only within the Space Link Subnetwork for special applications such as audio, video, high rate payloads, tape playback, and the intermediate transfer of Path and Internet data. Our interest is with the services provided by the SLS.

B. Space Link Subnet Services

The SLS supports the bidirectional transmission of data through the space/ground and space/space channels which interconnect the distributed elements of the CCSDS Advanced Orbiting Systems. It also provides "direct connect" transmission services for certain types of data which requires timely or high-rate access to the space channel. During SLS transfer, different flows of data are separated into different Virtual Channels, based on data handling requirements at the destination. These Virtual Channels are interleaved onto the physical channel as a serial symbol stream. A particular Virtual Channel may contain either packetized or bitstream data, or a combination of both.

The SLS consists of two layers; the "Space Link Layer" and the "Space Channel Layer" which correspond to ISO-equivalent Data Link layer and Physical layer respectively. Efficient use of the physical space channel was a primary driver in the development of these protocols. The Space Link layer is composed of the "Virtual Channel Link Control" (VCLC) sublayer and the "Virtual Channel Access" (VCA) sublayer.

The main function of VCLC sublayer is to convert incoming data into a protocol data unit which is suitable for transmission over the physical space channel. Four type of protocol data units may be generated by the VCLC sublayer: fixed length blocks of CCSDS Packets, called "Multiplexing Protocol Data Units" (M-PDUs); fixed length blocks of Bitstream data, called "Bitstream Protocol Data Units" (B-PDUs); fixed length blocks of mixed packetized and isochronous data, called "Insert Protocol Data Units" (IN-PDUs); and fixed length blocks of data for use by retransmission control procedures, called "Space Link ARQ Procedure Protocol Data Units" (SLAP-PDUs).

There are several procedures in VCLC sublayer to perform the function. The Encapsulation procedure provides the flexibility to handle virtually any packet structure. It puts a primary header to delimited data units (including Internet packet) and make it to be a CCSDS Packet. Multiplexing procedure multiplexes those CCSDS Packets on the same virtual channel together. The length of multiplexing protocol data unit is fixed since it is required to fit exactly in the fixed length data space of VCDU/CVCDU. There maybe some packets which overlap two or more M-PDU and "first packet pointer" points out where the first packet starts. Some user data, such as audio, video, playback and encrypted information, will simply be presented to the SLS as a stream of bits or octets. The Bitstream procedure simply blocks these data into individual Virtual Channels and transmits it. When the transmission rate is high, Bitstream data may be transmitted over a dedicated Virtual Channel. Alternatively, if the transmission rate is low, it can be inserted at the front of other packetized or bitstream data. This is called the Insert procedure. Through this procedure, bandwidth can be used more efficiently.

The last procedure of VCLC is Space Link ARQ Procedure (SLAP) which is used to provide guaranteed Grade-1 delivery of data links. The SLAP-PDU carries "Link ARQ Control Words" (LACWs) which report progress on receipt of data flowing in the opposite direction. Upon arrival at the receiving end, the LACW is extracted from the PDU, and the sequence number is checked to assure that no data has been lost or duplicated. In the event of a sequence error, the LACW carried by PDUs traveling in the opposite direction is used to signal that a retransmission is required. This retransmission begins with the first PDU that was not received in sequence, and all subsequent PDUs are retransmitted in the order in which they were originally provided to the LSAP from the layer above.

The VCA sublayer creates the protocol data units used for space link data transfer: these are either "Virtual Channel Data Units" (VCDUs) or "Coded Virtual Channel Data Units" (CVCDUs), and are formed by appending fixed length Header, Trailer and (for CVCDUs) error correction fields to the fixed length data units generated by the VCLC sublayer. The VCA sublayer is composed of "Virtual Channel Access" (VCA) and "Physical Channel Access" (PCA) procedures. VCA procedure generates VCDU for protocol data units which come from VCLC sublayer or accepts independently generated VCDU from reliable users. A VCDU with a powerful outer code of error-correcting Reed-Solomon check symbols appended to it is called a CVCDU. Relative to a VCDU, a CVCDU contains more error-control information and, hence, less user data. "Virtual Channel ID" which is field of VCDU/CVCDU Header can enable up to 64 VCs to be run concurrently for each assigned Spacecraft ID on a particular physical space channel. Since space data is transmitted through weak signal, noisy channel as serial symbol stream, a robust frame synchronization process at the receiving end is required. Therefore, fixed length VCDU/CVCDU is used and PCA procedure prefixes a 32 bits Synchronization Marker in front of VCDU/CVCDU to form a "Channel Access Data Unit" (CADU). A contiguous and continuous stream of fixed length CADUs, known as a "Physical Channel Access Protocol Data Unit" (PCA-PDU) is transmitted as individual channel symbols through the ISO-equivalent Physical Layer of the SLS, which is known as the "Space Channel Layer".

C. Space Link "Grades of Service"

Three different "grades of services" are provided by the SLS, using a combination of error detection, error correction and retransmission control techniques. We have to note that each VC can only support a single grade of service.

C.1 Grade-3 Service

This service provides the lowest quality of service. Data transmitted using Grade-3 service may be incomplete and there is a moderate probability that errors induced by the SLS are present and that the sequence of data units is not preserved. A VCUD is discarded if an uncorrectable error is detected at the destination. Grade-3 service should not be used for transmission of asynchronous packetized data, because it provides insufficient protection for the extensive control information contained in the packet headers.

C.2 Grade-2 Service

CVCDU is the unit of transmission that supports Grade-2 service. The Reed-Solomon encoding provides extremely powerful error correction capabilities. Data transmitted using Grade-2 service may be incomplete, but data sequencing is preserved and there is a very high probability that no data errors have been induced by the SLS.

C.3 Grade-1 Service

Data transmitted using Grade-1 service are delivered through the SLS complete, in sequence, without duplication and with a very high probability of containing no errors. It is provided by using two paired Reed-Solomon encoded Virtual Channels, in opposite directions, so that an Automatic Repeat Queueing (ARQ) retransmission scheme may be implemented.

IV. Simulation Results

The ADTV system described above without the priority and transport processors was simulated in detail. The simulation programs were written in *C* and implemented on a *SUN* workstation. Along with the ADTV system we also simulated a video coding

	Mean Rate (Mbits/sec)	STDR	PAR	Average PSNR	STDP	Subjective Test
Sequence 1	0.95	0.087	1.31	37.84	1.17	blurred, annoying blocks in neck
Sequence 2	1.91	0.068	1.12	41.25	0.94	very good
Sequence 3	0.95	0.091	1.31	37.93	1.12	blocky in neck, slight artifacts in background
Sequence 4	0.95	0.029	1.31	38.93	0.72	good except blocky first frame
Sequence 5	1.91	0.025	1.12	41.55	0.49	very good
Sequence 6	0.95	0.030	1.31	38.35	0.49	blocky first frame, blurred in neck, MC artifacts
Sequence 7	1.04	0.612	3.24	38.62	1.72	slight grain, better than Seq. 1 overall
Sequence 8	1.91	0.437	1.77	41.28	1.21	very good
Sequence 9	1.04	0.621	3.25	38.66	1.62	slightly blocky in neck

STDR: Standard deviation of coding rate(Mbits/sec)

PAR: Peak to average ratio

STDP: Standard deviation of PSNR(dB)

Sequence 1: ADTV, p=15, t=1   Sequence 2: ADTV, p=30, t=1

Sequence 3: ADTV, p=15, t=3   Sequence 4: H.261, p=15, t=1

Sequence 5: H.261, p=30, t=1   Sequence 6: H.261, p=15, t=3

Sequence 7: ADTV, p=15, t=1, QS=4 for INTRA frame

Sequence 8: ADTV, p=30, t=1, QS=4 for INTRA frame

Sequence 9: ADTV, p=15, t=3, QS=4 for INTRA frame

p: Bandwidth = px64 kbits/sec

t: Coding threshold after motion compensation (mean error)

Act. video pixels: (Luma) 352Hx240V, (Chroma) 176Hx120V, 30 frames/sec

Table 1 Performance of coding rate and PSNR using ADTV and H.261 coding schemes.

scheme based on the CCITT H.261 recommendations. The purpose was to have a benchmark for simulation.

In our ADTV simulator, the frames were arranged in the following sequence

I B B P B B P B B P B B I B B P...

The sequence used for testing the simulator was the *Susie* sequence. This sequence contains both low and moderate motion of the type to be encountered in most NASA applications. We present the results in the form of graphs, tables and a videotape accompanying this report.

The coding rates and PSNR under different coding conditions are listed in Table 1. Two parameters are used to control the coding conditions. The parameter p controls the output rate and length of the rate buffer. The fullness of the rate buffer determines the quantizer stepsize and therefore, the coding rate and quality. Thus the parameter p has an important impact on both coding rate and quality. The parameter t is used to decide whether the macroblock after motion compensation needs coding. Smaller values of t will lead to higher rates, while smaller values of t will also result in higher quality.

The first two sequences were coded using the ADTV algorithm. In Sequence 1, with p = 15, the average rate is 0.37 bits/pixel. This rate is not sufficient to effectively code the I frames. As the B and P frames depend heavily on the I frames, poorly coded I frames create annoying blocking effect which cascades down the entire sequence. As the quantizer stepsize depends on how full the buffer is, this low rate leads to the buffer getting filled up as the lower portions of the I frame are being coded. This means that when coding the lower regions of the I frames, the quantizer is coarse. This results in blocking artifacts which are very noticeable in the lower portions of the sequence. This effect can be seen in the first sequence on the videotape. When the coder has finished with the I frame, and the B and P frames are being coded, as the coding rate is lower for the B and P frames, the buffer situation gets partly remedied. However, this is not sufficient to get the quantizer stepsize small enough to remove the blocking effect. When the buffer size is increased (p = 30) the number of artifacts is reduced as seen in the second sequence on the video tape. An interesting effect can be seen in the third sequence. Here the buffer size was kept the same as in the second sequence, however, the motion compensation threshold t was kept high. This means that blocks that would have been coded in Sequence 2 are left uncoded in Sequence 3. This in turn emphasizes the blocking effects. One would think that given the fact that we are accepting more distortion, the rate would go down. However as we can see from Figure 1, the rates for Sequence 2 and Sequence 3 are almost identical. This could be attributable to the fact that the poor reconstruction of the P frames lead to poor prediction and hence an increase in bit rate which takes away any savings from the higher motion compensation threshold. Thus the parameter t while it affects the quality (creating artifacts in smooth background) has little effect on the rate.

From Figure 1 we can see that the ADTV algorithm generates a very bursty traffic. This is in sharp contrast to the CCITT H.261 algorithm which produces relatively smooth output. We can see this from the results for Sequences 4-7 which were coded using the CCITT H.261 algorithm using the same parameters p and t as the first three sequences. The rate and PSNR results for these sequences are given in Table 1 and Figures 3 and 4. Recall that the only significant difference between the CCITT H.261 algorithm and the ADTV algorithm are the sequencing and motion compensation. The use of the intra-frame coding every 13th frame in the ADTV algorithm increases the bit rate. This is compensated for by using the different motion compensation approach giving the bursty traffic. In the CCITT H.261 algorithm, intra-frame coding is recommended but a macroblock must be updated at least once every 132 times it is transmitted. Thus there is no significant variation in the rate from frame to frame. The disadvantage of the CCITT H.261 algorithm when compared to the ADTV algorithm is the decrease in the ability to randomly access any particular frame, and the decrease in the ability to react fast to sudden scene changes. Furthermore it should be noted that the ADTV algorithm was proposed for HDTV sequences, while the sequences we are using have significantly less resolution.

Due to the importance of I frame, which serves as the anchor frame for both P and B frame, we decide to put more coding efforts in such frame to try to eliminate the blocking effect. In Sequences 7-9, the ADTV algorithm has been modified to keep the quantization stepsize QS constant while coding the I frame. One effect is that the buffer becomes really full during coding the I frame, and the subsequent frame gets very little of the coding resources. This results in an increase in burstiness as can be seen from Figures 5 and 6. However, this approach does result in the reduction/elimination of the blocking effect. That such a simple strategy can result in such dramatic improvement shows that should blocking effects appear in the HDTV sequences, attention should be paid to the encoding of the I frames.

Figures 7-14 show various comparison results between the ADTV, the modified ADTV and the CCITT H.261 algorithms. While these comparisons show an advantage for the CCITT H.261 algorithm, subjective comparisons tend to show the reverse. We invite the reader to examine the videotape and draw their own conclusions.

V. HDTV Transmission on the CCSDS Network

As described in the previous section, some user data, such as audio, video, playback and encrypted information, can simply be presented to the Space Link Subnet (SLS) as a stream of bits or octets. The SLS merely blocks these data into individual Virtual Channels and transmits them using Bitstream Service. Some bitstream data, such as digitized video and audio, will have stringent delivery timing requirement and are known as "isochronous" data. For the transmission of ADTV coded information, the channel transmission rate is high enough to dedicate a specific Virtual Channel. Although the coding output rate is quite bursty, there are two mitigating circumstances

1. the pattern of burstiness is relatively "uniform". That is, the data rate peaks every 13th frame.

2. the variations occur very fast, that is high traffic persists for only a single frame followed by low traffic.

Because of (2) the traffic can be smoothed out using a moderate sized buffer, and (1) implies that the size of the buffer can be ascertained with some confidence.

The delay constraints on the transmission preclude the use of the Space Link ARQ procedure, while the delay constraint coupled with the high rate argue against the use of the Insert and Multiplexing procedures. This leaves the Bitstream procedure as the only viable candidate for HDTV transmission. This conclusion coincides with that of the CCSDS Red Book *Audio, Video, and Still-Image Communication Services*[3].

The Bitstream service fills the data field of the B-PDU (Bitstream-Protocol Data Unit) with the Bitstream data supplied at user's request. Each B-PDU contains data for only one VC, identified by the VCDU-ID parameter. Each bit is placed sequentially, and unchanged, into the B-PDU data field. When the Bitstream data have filled one particular B- PDU, the continuation of the Bitstream data is placed in the next B-PDU on the same VC. Due to the delay constraints of the PDU release algorithm, if a B-PDU is not completely filled by Bitstream data at release time, some fill pattern has to be filled into the remainder of the B-PDU.

As far as the grade of service is concerned, one could use the error protection service provided by the ADTV algorithm with grade 3 service, or discard any error protection from the ADTV signal and use grade-2 service. Given the sketchy amount of information available about forward error correction in the ADTV algorithm we would suggest the use of the Grade-2 service in the CCSDS recommendations. Some kind of forward error correction is imperative because of the need for data sequencing along with the general need for video integrity. Therefore, Grade-2 service which adopts Reed-Solomon encoding is a logical choice. Recall that priority layer in ADTV system distinguishes cell into High/Low priority depending on its importance for video reconstruction. In CCSDS packet, no corresponding priority bit is allocated in the header and all cells are equally treated. Another possible alternative for transmitting ADTV signal over CCSDS network is reserving two virtual channels, one with Grade-2 service for high priority cell and another with Grade-3 service for low priority cell. Whether it is more efficient needs further works. According to the minimum predicted performance of Grade-2 service[4], the probability that a Coded Virtual Channel Data Unit (CVCDU) will be missing is 10^{-7}. If we assume a CVCDU contains 8800 bits of data, from our simulation, about 95 macroblocks of video data (for ADTV format, a frame is formed by 90Hx60V macroblocks) will get lost in a duration slightly over one and half hour. This shouldn't hurt the quality too much in motion compensation scheme. The probability that a CVCDU contains an undetected bit error is 10^{-12}, only one bit error will occur in a transmission period over 11 hours. By the way, error concealment scheme which is performed at the receiver alone can be used to detect and repair the damaged portions of the image. Therefore, if this error bit occurs in video data, it won't be easy to notice the degradation. But if the error bit occurs in control data, some degree of damage is inevitable. It may therefore be desirable to provide some more protection to the control data before it enters the network. We are still looking at this particular issue.

VI. Conclusion

In this paper, the basic principals of ADTV coding scheme and CCSDS Principal Network are described. We also present the simulation results for ADTV system and compare it with H.261 coding scheme. Based on the simulation results, we investigate the possibility of transmitting a HDTV-like signal over CCSDS network. Some suitable approaches have been suggested. The sequences we used for testing the simulator and generating the results are those used for testing the MPEG algorithm. The sequences are of much lower resolution than the HDTV sequences would be, and therefore the extrapolations are not totally accurate. We would expect to get significantly higher compression in terms of bits per pixel with sequences that are of higher resolution. However, the simulator itself is a valid one, and should HDTV sequences become available, they could be used directly with the simulator.

References

[1] *Advanced Digital Television System Description*, The Advanced Television Research Consortium, Feb. 1991.

[2] K. Joseph, S. Ng, R. Siracusa, D. Raychaudhuri, and J. Zdepski, "Prioritization and transport in the ADTV digital simulcast system," *IEEE Trans. Cosumer Electronics*, vol. 38, no. 3, pp. 319-323, Aug. 1992.

[3] CCSDS Red Book, *Audio, Video, and Still-Image Communication Services*,?.

[4] CCSDS Green Book, *Summary of Concept, Rationale and Performance*, October 1989.

[5] H. Sun, K. Challapali, and J. Zdepski, "Error

concealment in digital simulcast AD-HDTV decoder," *IEEE Trans. Consumer Electronics*, vol. 38, no. 3, pp. 108-117, Aug. 1992.

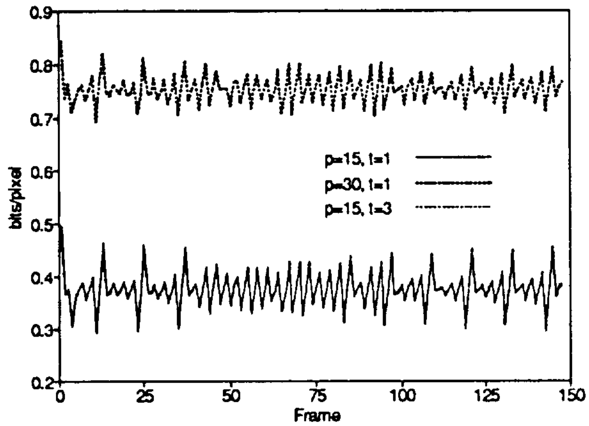


Figure 1 Coding rate using ADTV coding scheme.

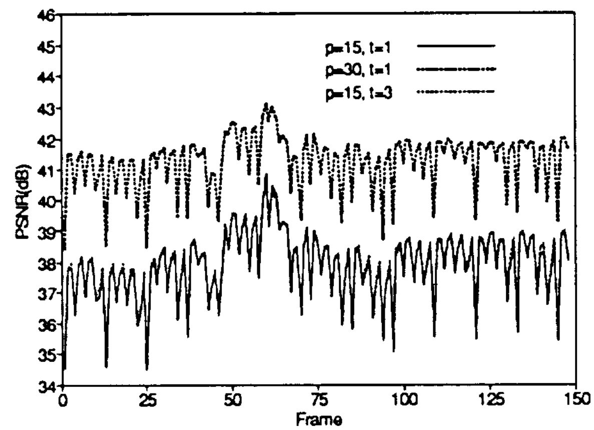


Figure 2 PSNR using ADTV coding scheme.

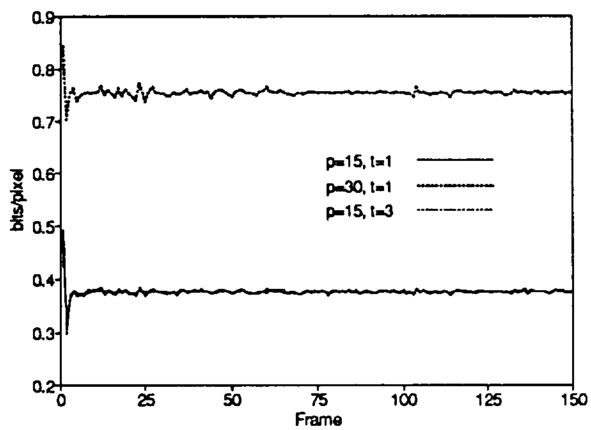


Figure 3 Coding rate using H.261 coding scheme.

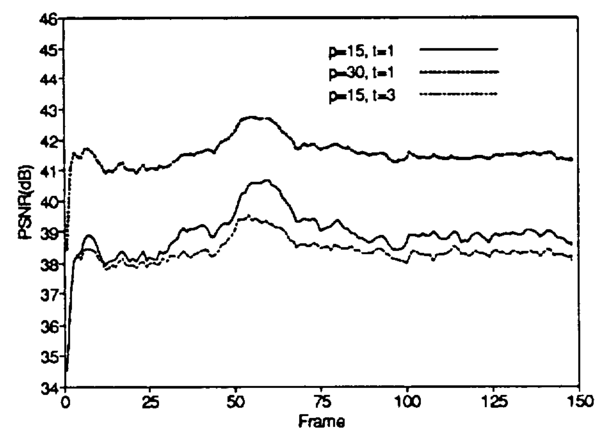


Figure 4 PSNR using H.261 coding scheme.

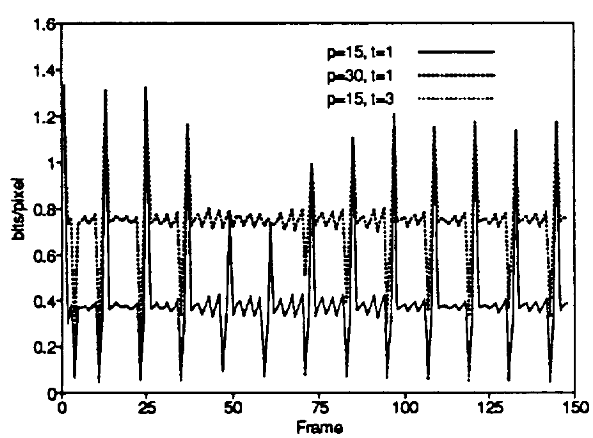


Figure 5 Coding rate using modified ADTV coding scheme.

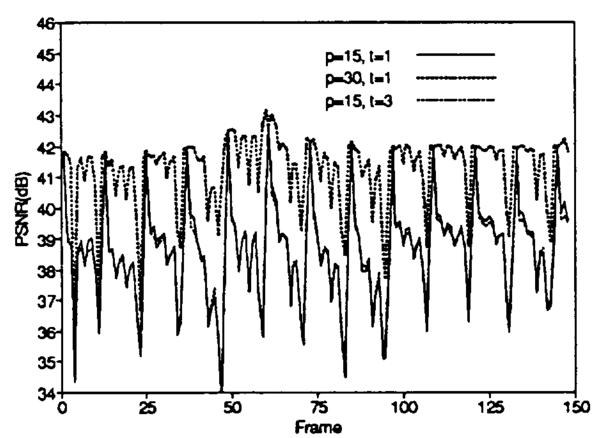


Figure 6 PSNR using modified ADTV coding scheme.

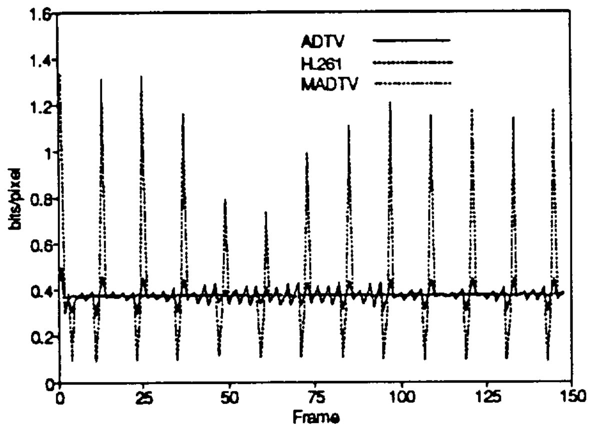


Figure 7 Comparison of coding rate (p=15, t=1).

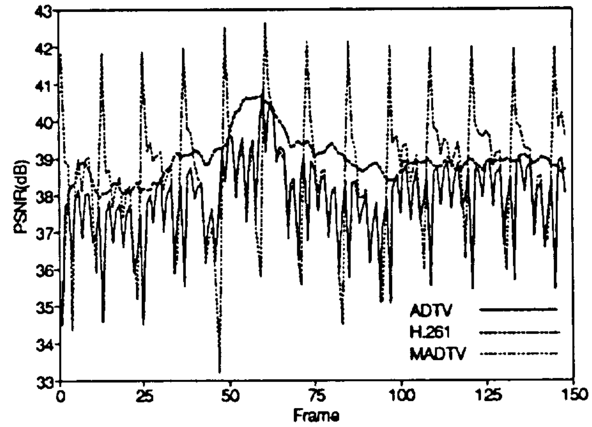


Figure 8 Comparison of PSNR (p=15, t=1).

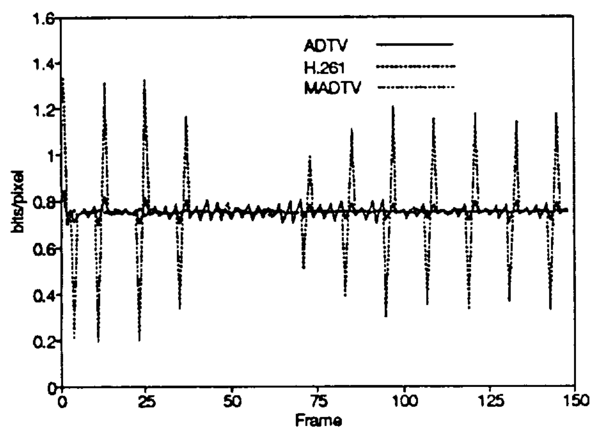


Figure 9 Comparison of coding rate (p=30, t=1).

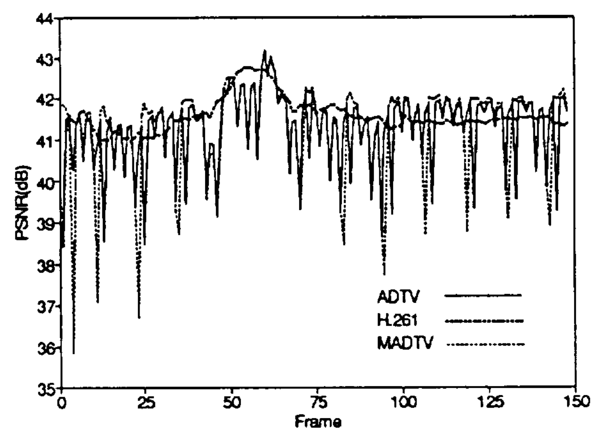


Figure 10 Comparison of PSNR (p=30, t=1).

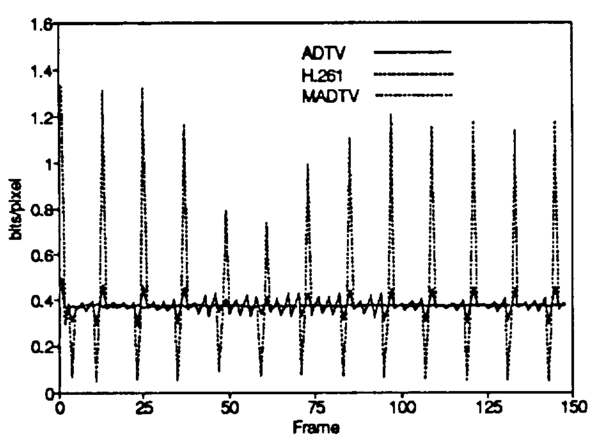


Figure 11 Comparison of coding rate (p=15, t=3).

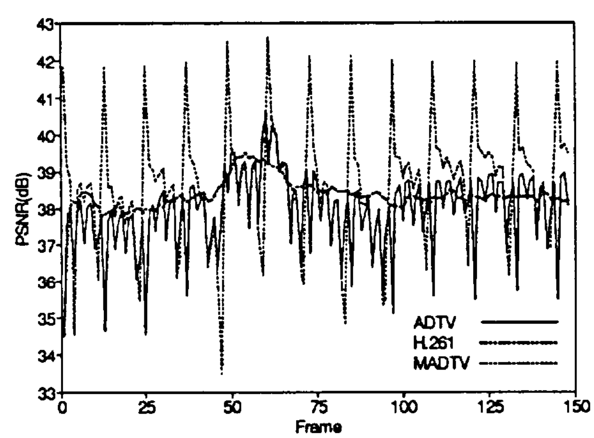


Figure 12 Comparison of PSNR (p=15, t=3).